Universidad Carlos III de Madrid

Escuela Politécnica Superior

Ingeniería Técnica en Informática de Gestión

Proyecto Fin de carrera

Análisis y Clasificación de las Tecnologías, Técnicas y Herramientas de Big Data

Autor: Juan Antonio Ayuso Rodríguez

Tutor: Ángel Lagares Lemos

Índice

1. INTRODUCCIÓN

1.1.ABSTRACT

La información siempre ha sido un importante activo, valioso para países, organizaciones, empresas, etc… Históricamente, aquellos que han tenido más información, disfrutaban de una ventaja competitiva, que la mayor parte de las veces resultaba definitiva.

En un mundo competitivo como el de hoy, tener esta ventaja es, si cabe, aún más importante para poder lograr tus objetivos y estar por encima de tus competidores.

En los últimos años la velocidad a la que se genera la información en el mundo está creciendo exponencialmente y esto implicaba dos problemas para los que se tenía que buscar solución. El primero era lograr almacenar toda esta información, y el segundo, poder analizarla, interpretarla y usarla en beneficio propio.

Para resolver el primer problema, es sencillo darse cuenta de que junto con el aumento del volumen de información generado en el mundo, también ha ido evolucionando, más o menos a la misma velocidad, la tecnología que permite almacenarla. Cada vez aparecían dispositivos o tecnologías de almacenamiento con mayor capacidad y a menor coste.

En cuanto al segundo problema, también es evidente que la tecnología de procesamiento de esta información ha ido evolucionando, aumentando la capacidad de proceso y disminuyendo el tiempo necesario para realizarlo, sin embargo, llegaba un punto en el que la cantidad de datos a analizar era tal que el tiempo que se necesitaba para procesarla provocaba que en el momento en el que se obtenían los resultados, la mayor parte de las veces, esta información estaba obsoleta, había cambiado y por lo tanto ya no era útil.

Esto podía pasar por dos razones: una era la necesidad de procesar tal cantidad de información que el tiempo de proceso era muy alto (a veces podría llegar a meses, incluso años), mientras que la otra, aunque la cantidad de información que se necesitase procesar no fuera tan grande, en ocasiones existía la necesidad de obtener el resultado en tiempo real, en algunos casos obtener el resultado del análisis de la información en diez segundos no es suficiente.

En este contexto surge, para responder a esta necesidad, lo que comúnmente se llama Big Data, sistemas que se encargan de gestionar la gran cantidad de información que se genera hoy en día, y sacar provecho de ella.

Debido al rápido avance de estas tecnologías que, tal y como se ha comentado en el punto anterior, se encargan de gestionar de forma óptima la gran cantidad de información que se genera en el mundo, aparece la necesidad de realizar un análisis completo del ecosistema del Big Data, aclarando los conceptos, modelos, técnicas y herramientas que se están usando actualmente. Este análisis del ecosistema del mundo de Big Data es necesario para permitir la exploración efectiva, la comprensión, evaluación, comparación y selección de metodologías, lenguajes, técnicas, plataformas y herramientas.

En este proyecto de fin de carrera, se revisa el estado del arte del Big Data desde una perspectiva holística. Se realiza un análisis detallado de uno de los landscape más típico que se pueden encontrar, aprovechando para realizar un estudio minucioso sobre las diferentes opciones tecnológicas existentes. Y por último, se realiza también la creación de un landscape alternativo, que permite realizar una clasificación diferente y nueva, que intenta ayudar, no solo a entender mejor el mundo del Big Data, sino también a poder tomar la mejor decisión en una situación en la que se debe optar por un sistema Big Data u otro, gracias al conocimiento de las diferentes alternativas, de cara a optimizar los resultados obtenidos por el sistema.

1.2. OBJETIVOS Y ESTRUCTURA DEL DOCUMENTO

El contenido del presente documento ha sido estructurado de la siguiente forma:

1. **Introducción**. En este tema se describe la situación que ha dado lugar al desarrollo de este proyecto y se establecen los principales objetivos del mismo.
2. **Estado del Arte**. En este tema se analiza el panorama actual del mundo del big data, introduciendo los principales conceptos para su entendimiento y realizando un paseo por toda su historia desde el comienzo hasta los días actuales.
3. **Big Data Landcape**. En este tema se analiza uno de los landscape existentes y a través de él, se describen cuáles son las principales arquitecturas en las que se basa un sistema de Big Data.
4. **Nuevo Landscape Big Data**. En este tema se intenta realizar un landscape original, una clasificación mediante la cual se pueda mejorar el entendimiento de este mundo, el uso de sus herramientas y las diferentes posibilidades que pueden ofrecer.
5. **Conclusiones**. En este tema se reflexiona sobre los objetivos alcanzados con este proyecto.
6. **Presupuesto**. En este punto se calcula el coste del proyecto, teniendo en cuenta todos los elementos que han sido necesarios para realizarlo.
7. **Referencias y Bibliografía**. Por último se enumeran las diferentes fuentes de información que han ayudado en la elaboración de este proyecto.

2. ESTADO DEL ARTE

2.1. DEFINICIÓN

Si se intenta buscar una definición de big data en internet encuentras varias como:

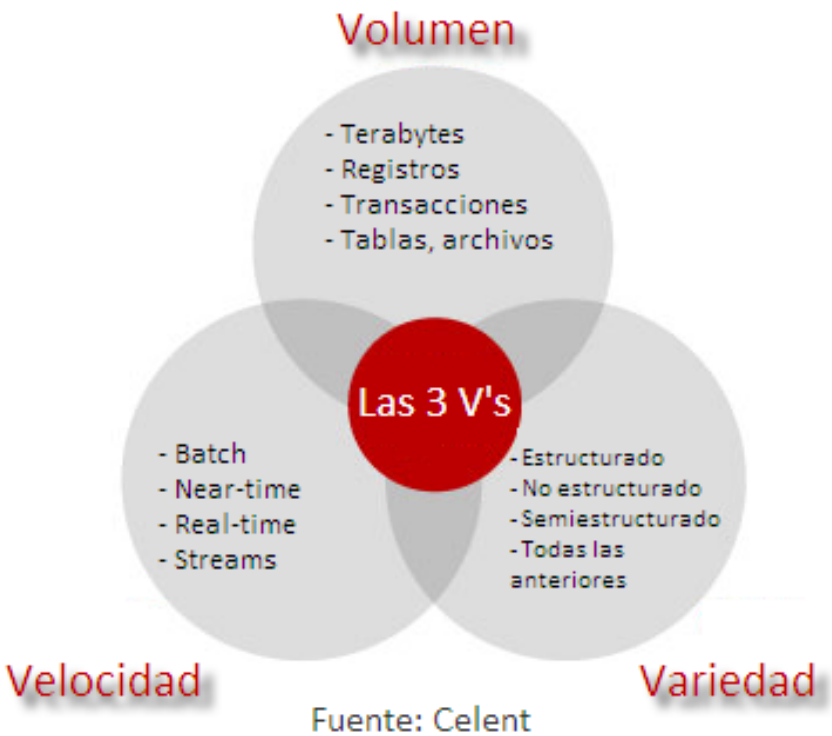
- ✓ "Es el sector emergente del área de las tecnologías de la información y la comunicación que se preocupa de cómo tratar y almacenar grandes cantidades de información o conjuntos de datos."
- ✓ "Es un término aplicado a conjuntos de datos que superan la capacidad del software habitual para ser capturados, gestionados y procesados en un tiempo razonable. Los tamaños del 'big data' se hallan constantemente en aumento."

Estas definiciones se centran sobre todo en relacionar 'big data' con el volumen de datos, sin embargo, una posible definición más completa, puede ser la que dio Edd Dumbill:

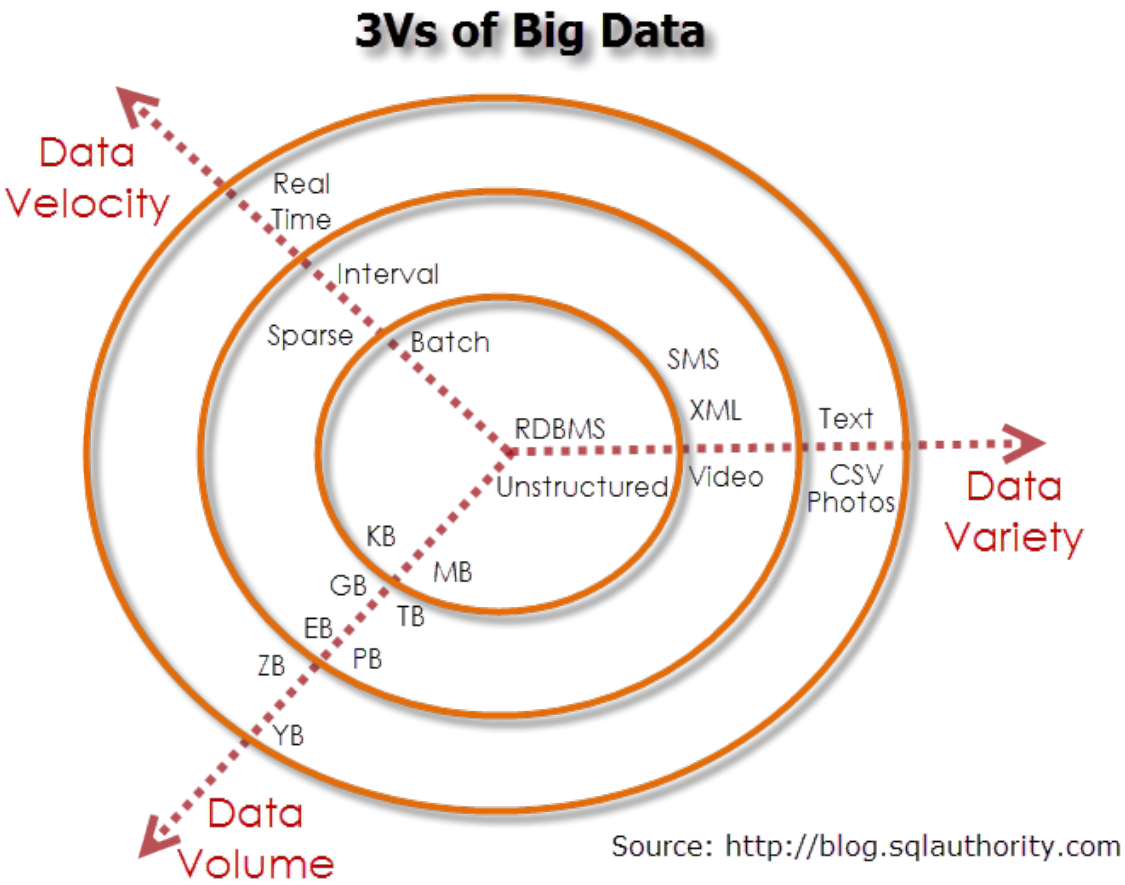
- ✓ "Big Data son aquellos datos que sobrepasan la capacidad de proceso de las bases de datos convencionales. Los datos son demasiado grandes, se mueven demasiado rápido o no encajan en las estructuras de las arquitecturas de bases de datos tradicionales. Para conseguir un valor añadido de todos estos datos, se debe elegir una forma alternativa de procesar esta información"[1]

En ella, introduce conceptos como la velocidad a la que se mueven los datos o la estructura que tienen.

Hoy en día, prácticamente todo el mundo define Big Data a través de lo que se conoce como las tres 'v'. Pero, ¿Cuáles son las tres 'v'? A continuación se muestran dos gráficos que responden a esta pregunta.



Fuente: Celent



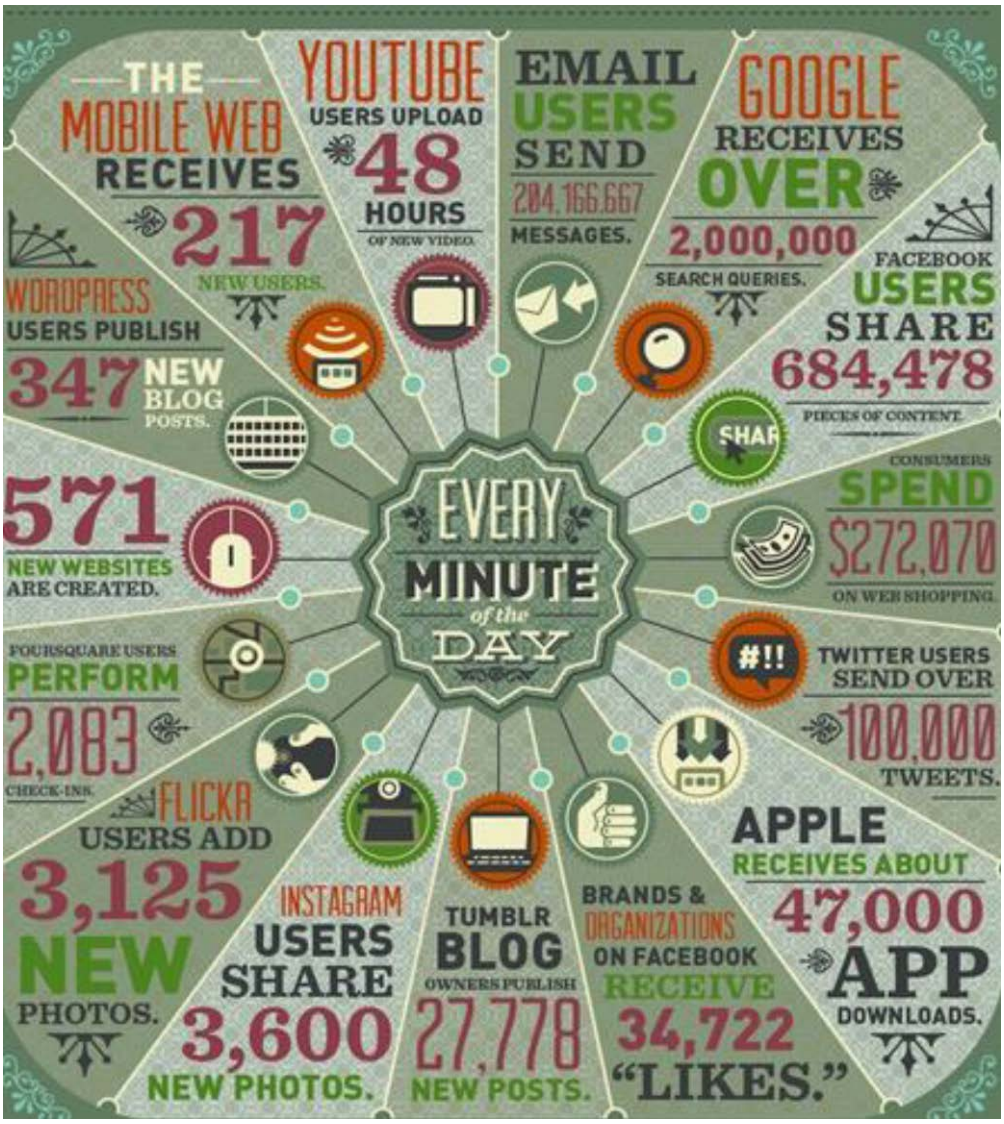
Source: http://blog.sqlauthority.com

2.1.1. Volumen.

En la actualidad nos enfrentamos a un crecimiento exponencial de la cantidad de datos que se generan en el mundo. El gran avance de la tecnología ha abierto las puertas hacia un nuevo entendimiento y toma de decisiones. Esta tecnología nos ayuda a entender y describir esta gran cantidad de datos que tomaría demasiado tiempo y sería demasiado costoso cargarlos en una base de datos relacional para su análisis. El concepto 'big data' se refiere por tanto a todo aquello que nos ayuda a procesar y analizar toda esta gran cantidad de información que no puede ser procesada y analizada con herramientas tradicionales.

Para comprender un poco mejor este gran crecimiento en la cantidad de datos que se generan, a continuación se enumeran varias estadísticas curiosas:

- ✓ En un solo día de 2011 se produjeron más datos que en todo el año 2000.
- ✓ Al día se generan 2,5 trillones de datos. El 90% de los datos que existen hoy en día se han generado en los dos últimos años.
- ✓ Se crean más de 571 nuevas páginas web cada minuto.
- ✓ El tráfico de datos móviles creció en 2013 un 81% y es equivalente a 18 veces todo el tamaño de internet en el año 2000.
- ✓ Cada minuto se suben 30 horas de video, 3.000 fotos nuevas, se envían más de 204 millones de mails y se realizan más de 2 millones de búsquedas.
- ✓ Se envían cerca de 340 millones de tweets cada día, unos 4.000 por segundo.
- ✓ Facebook tiene más de 901 millones de usuarios que generan datos de interacción social. Cada minuto se visitan más de 6 millones de perfiles en Facebook.
- ✓ Cada segundo se registran en el mundo más de 10.000 transacciones de pagos con tarjetas.
- ✓ Más de 5.000 millones de personas llaman, mandan un mensaje de texto, crean un tweet o navegan por internet con teléfonos móviles.
- ✓ Se estima que en el año 2020 se produzcan 44 veces más datos que los que se generaron en el año 2009.



2.1.2. Variedad.

La variedad de los datos en el mundo ha explotado, pasando de ser datos estructurados, almacenados en bancos de datos empresariales, a ser desestructurados, semi-desestructurados, datos de audio, datos de video, XMLs, etc…

Existen fuentes cada vez más numerosas y diferentes que generan datos que pueden ser útiles si se pudieran procesar como por ejemplo:

- ✓ Datos en streaming
- ✓ Cotizaciones bursátiles
- ✓ Medios sociales
- ✓ Datos máquina a máquina

- ✓ Datos de sensores
- ✓ Dispositivos móviles

Estas fuentes generan, cada vez más, una creciente variedad de datos que son susceptibles de ser procesados y convertidos en información. Datos de tipo texto, audio, imágenes, vídeo, tweets, etc…

Esta creciente variedad de los datos viene como consecuencia de la mayor conectividad a nivel global en el mundo. A esto ha contribuido en gran medida la proliferación de los smartphone, que ya han superado con creces los 1.000 millones de dispositivos en el mundo y que se prevé que para 2019 llegue a más de 5.500 millones multiplicando por diez el tráfico de datos que generan (unos 10 Exabytes)[2].

Otro concepto que ha ayudado a este aumento de la variedad en los datos es lo que se denomina el de 'el internet de las cosas' [3], objetos que se conectan a internet generando datos a través de sensores. Sensores de estaciones meteorológicas, de automóviles, de electrodomésticos (domótica), son ejemplos de objetos que se conectan a internet e interactúan en él, creando información a través de sus sensores.

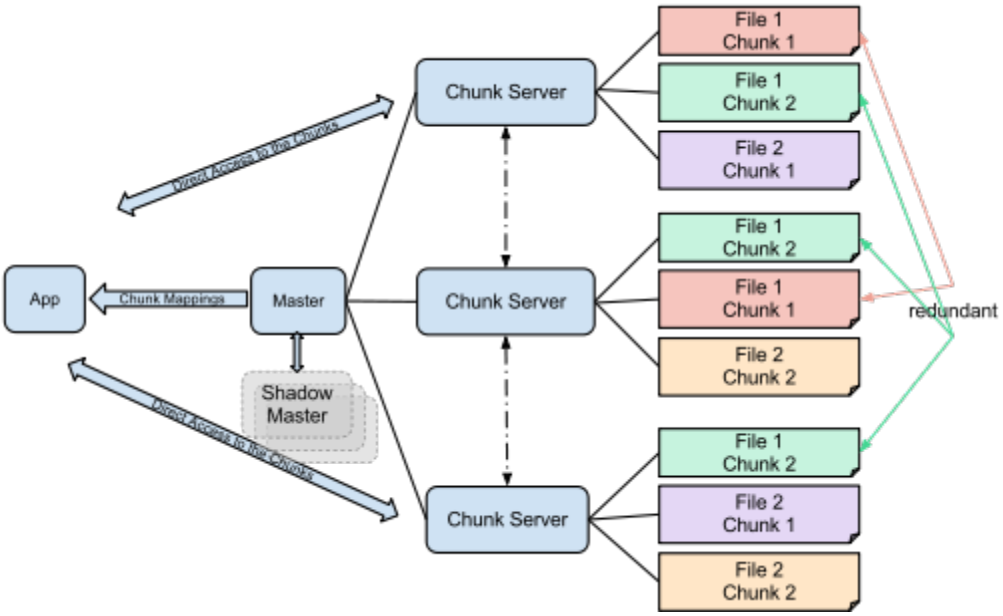
2.1.3. Velocidad.

La velocidad a la que se están creando los datos también está creciendo exponencialmente. Este ritmo con el que se generan los datos desde diversas fuentes, como pueden ser procesos de negocio (sistemas informáticos de empresas o bancos), los sensores de las máquinas, las redes y la interacción del hombre con medios sociales o con dispositivos móviles, es masivo y continuo. El análisis de estos datos en tiempo real puede ayudar a empresas o investigadores a tomar decisiones que pueden proporcionar una ventaja competitiva.

2.2. UN POCO DE HISTORIA

Se puede decir que el origen de esta nueva tecnología se debe a google en 2003 cuando publica un artículo sobre Google File System (GFS)[4]. GFS es un sistema de archivos distribuido y escalable que proporciona tolerancia a fallos aún ejecutándose en hardware de bajo coste.

La idea básica era poder almacenar información de forma fiable incluso en máquinas cuya fiabilidad no era segura. Esto lo conseguían almacenando como mínimo tres copias de cada fichero en un servidor cluster determinado. Esta redundancia lo hacía resistente a posibles errores de memoria, de conexión, fallos de disco, etc…



Por otro lado, para gestionar de manera eficiente el uso del espacio de almacenamiento de datos, como en general sus archivos tenían un tamaño que iba desde los 100Mb hasta varios Gb, organizaron la memoria en bloques (en inglés se usa el término “chunk”) de 64 Mb. Por ejemplo, para un fichero de 128Mb se tenían que usar dos bloques para almacenarlo, si el fichero fuera de 1Mb, se usaría un bloque entero de 64Mb para almacenarlo quedando prácticamente entero vacío, pero eso no les preocupaba porque, como se comentaba anteriormente, la mayor parte de los fichero eran mayores de 100Mb y los ficheros más comunes eran múltiplos de 64Mb.

Un cluster estaba compuesto por un servidor maestro y cientos o miles de servidores ("chunkservers") que son los que almacenan realmente los datos. En el servidor maestro están los metadatos tales como nombres de fichero, tamaños y ubicación de los mismos. Cuando un cliente intenta acceder a un fichero, el servidor maestro le proporciona la dirección de los servidores relevantes para recuperarlos. El servidor maestro está continuamente escuchando de todos los servidores que gestiona lo que se denomina "heartbeat". Si el heartbeat se detiene, el servidor maestro asigna inmediatamente otro servidor para que tome el relevo del que parece estar "roto".

Un año después, en 2004, Jeffrey Dean y Sanjay Ghemawat, de Google, publican su artículo sobre MapReduce[5]. Mapreduce es un modelo de programación y una aplicación asociada para el procesamiento y la generación de grandes cantidades de información.

Los programas escritos en este estilo funcional están paralelizados y ejecutados en un gran grupo de máquinas de forma automática. El sistema se encarga de detalles como partir los datos de entrada, la planificación de la ejecución del programa a través de un conjunto de máquinas, manejar y controlar los posibles fallos de la máquina, y la gestión de la comunicación entre las diferentes máquinas. Esto permite a los programadores sin experiencia en los sistemas paralelos y distribuidos utilizar fácilmente los recursos de un gran sistema distribuido.

En 2005, Doug Cutting y Mike Cafarella de Yahoo, aprovechando el trabajo de google desarrollan una herramienta que es clave para la explosión del concepto Big Data, Hadoop, que es un framework open source que se centra en el procesamiento de grandes cantidades de datos en sistemas distribuidos. Este framework utiliza Hadood Distribute File System que fue diseñado para almacenar grandes ficheros de forma fiable en varias máquinas dentro de un mismo cluster. Este sistema está basado en Google File System.

Posteriormente se cede a la fundación Apache y se empieza a distribuir en la versión 2.0.

2.3. DIFERENCIAS ENTRE BUSSINES INTELLIGENCE (BI) Y BIG DATA ANALYTICS[6]

Antes de empezar a comparar y a describir los pro y contra de cada uno de estos dos métodos de análisis de datos se debería responder a la pregunta ¿Realmente es necesaria la aparición del big data? Es decir, ¿cuál es la ventaja de analizar todos y cada uno de los datos y no solo muestras estadísticas tal y como se hacía hasta ahora?

Por ejemplo, al hacer una encuesta electoral, ¿sería necesario saber lo que piensa votar todo el censo? Si esto fuera así, la encuesta dejaría de ser una encuesta, serían directamente las elecciones.

Estadísticamente, existe una fórmula que en función del tamaño de la población y del nivel de confianza necesario y el margen de error permitido, te da el tamaño de la muestra ideal. La fórmula es

$$n = \frac{N \cdot z_{\frac{\alpha}{2}}^2 \cdot p \cdot (1-p)}{e^2 \cdot (N-1) + z_{\frac{\alpha}{2}}^2 \cdot p \cdot (1-p)}$$

Si el tamaño de la población tiende a infinito, el resultado es que el tamaño de la muestra ideal para que los datos sean representativos de la población con un nivel de confianza del 95% y un margen de error del 5% es de menos de 400

$$\lim_{N \to \infty} \frac{N \cdot (1.96)^2 \cdot 0.5 \cdot 0.5}{0.0025 \cdot (N-1) + (1.96)^2 \cdot 0.5 \cdot 0.5} = 384.16\ldots$$

Si esto es así, ¿por qué se va a necesitar, o que ventaja tendría poder analizar toda la información, si con tener la información de menos de 400 sujetos ya obtendríamos un resultado representativo?

Realmente hay muchas razones:

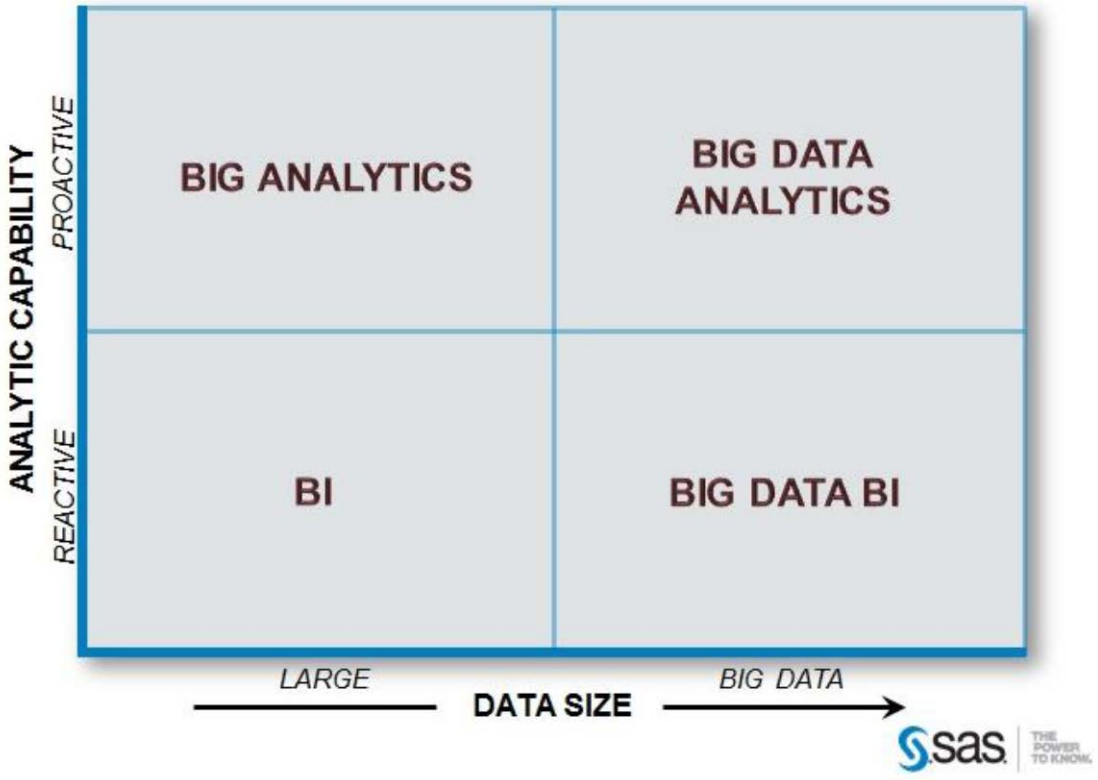
- ✓ Un buen resultado depende de tener una muestra heterogénea dentro de la población, cosa que a veces es complicado.
- ✓ En la estadística solo se tienen datos medios. Si se necesita analizar datos de un nicho de la población que no es el habitual (el medio), ¿Cómo accederíamos a sus datos si solo recogemos una muestra global?

- ✓ Puede que el 5% de margen de error no sea asumible. Por ejemplo, para estudios médicos en los que no se puedan permitir este margen de error.
- ✓ Si quisiéramos analizar los millones de transacciones que pasan por el servidor de un banco, no serviría analizar el comportamiento medio porque no nos daría información ya que cada transacción es única
- ✓ Lo mismo pasaría si se desean analizar las órdenes de compra venta de la bolsa. Para realizar un buen estudio, no serviría con analizar los comportamientos medios.

Por tanto, podemos ver que no todos los estudios dependen únicamente de la estadística, se deben tener en cuenta más factores, que a veces te obligan a poder analizar grandes cantidades de datos para poder tener unas conclusiones útiles.

Podemos considerar el análisis de big data como una evolución del análisis de datos con business intelligence, ¿quiere decir esto que viene a sustituirlo? En realidad no, los estudios de big data no excluyen a las herramientas de business intelligence tradicionales, ni a las bases de datos tradicionales, sino que son complementarios. Big data puede servir para cubrir huecos que las tecnologías anteriores no podían cubrir.

Para explicar mejor la diferencia entre una y otra opción, a continuación se muestra un gráfico hecho por SAS Institute[7] en el que la capacidad de análisis proactivo o reactivo está en la abscisa ´y´, mientras que en la abscisa ´x´ está el tamaño de los datos a analizar.



En el gráfico se pueden ver las cuatro principales soluciones de software que existen actualmente:

- ✓ Business intelligence (BI): Este escenario es aquel en el que necesitas gestionar grandes cantidades de datos y habilitar informes para que los usuarios finales puedan tener acceso a la información, ya sea de forma resumida o haciendo drill down hasta el detalle, lo ideal es contar con software de business intelligence, que proporciona una buena visión del rendimiento o cualquier otro aspecto de la empresa en el pasado. Este es el cuadrante inferior izquierdo del gráfico.
- ✓ Big Data BI: El segundo escenario se produce cuando la cantidad de datos se vuelve muy grande o las fuentes de los datos son externas o el tipo de dato que necesitas procesar no está estructurado o si a los usuarios les lleva demasiado tiempo conseguir la información que necesitan en un informe o al sistema no puede combinar fuentes de datos lo suficientemente rápido o en cualquier caso solo necesitas proporcionar análisis de datos reactivos. Este escenario es el más común actualmente en el mercado, y la mayor parte de las empresas lo están intentando resolver con soluciones basadas en SQL. Este es el cuadrante inferior derecho del gráfico.

- ✓ Big Analytics: Tal y como se ha comentado en los párrafos anteriores, se necesita otro tipo de análisis si lo que necesitas es tomar decisiones sobre el futuro, es decir, si necesitas conocer las preferencias de los clientes, o buscas la optimización de las rebajas o necesitas realizar predicciones sobre fraude, se necesita un tipo diferente de arquitectura. Este tipo de problemas implica normalmente un gran crecimiento del tamaño de los datos que se tienen que analizar y unos análisis proactivos de los mismos. El problema no es tanto que los datos sean tan grandes que tu sistema se vuelva lento para analizarlos, sino el hecho de que se tienen que realizar varias pasadas sobre ellos con análisis más complejos que podrían durar horas por cada pasada y realmente necesitas el resultado en minutos o segundos. Este escenario está representado en el cuadrante superior izquierdo del gráfico.
- ✓ Big Data Analytics: Ahora existen problemas en las empresas que tienen una cantidad de datos ingente y necesitan analizarlos proactivamente para tomar decisiones. Por ejemplo, empresas que tienen cientos de millones de SKU (stock Keeping Unit o número de referencia) repartidos en múltiples tiendas minoristas, o también futuras fuentes de datos, como los datos telemáticos generados por diversos sensores o indicadores en la industria (por ejemplo del automóvil) que pueden llegar a ser muy útiles tanto para los fabricantes, como para las aseguradoras. Son problemas con los que antes no se lidiaba, y que no son problemas de datos pequeños. Realmente no necesitas un resumen de esta información, lo que necesitas es poder predecir problemas en los sistemas de seguridad del coche antes de que puedan tener impacto tanto en los clientes como en las compañías aseguradoras que quieren ajustar sus tarifas para los conductores de menos riesgo. Este escenario de problemas es el que se representa en la esquina superior derecha del gráfico.

Tal y como se puede ver en esta comparación, no se trata de que unas tecnologías sean mejores que otras, sino que hay diferentes tipos de problemas que resolver y cada tecnología se ajusta mejor a unos que a otros.

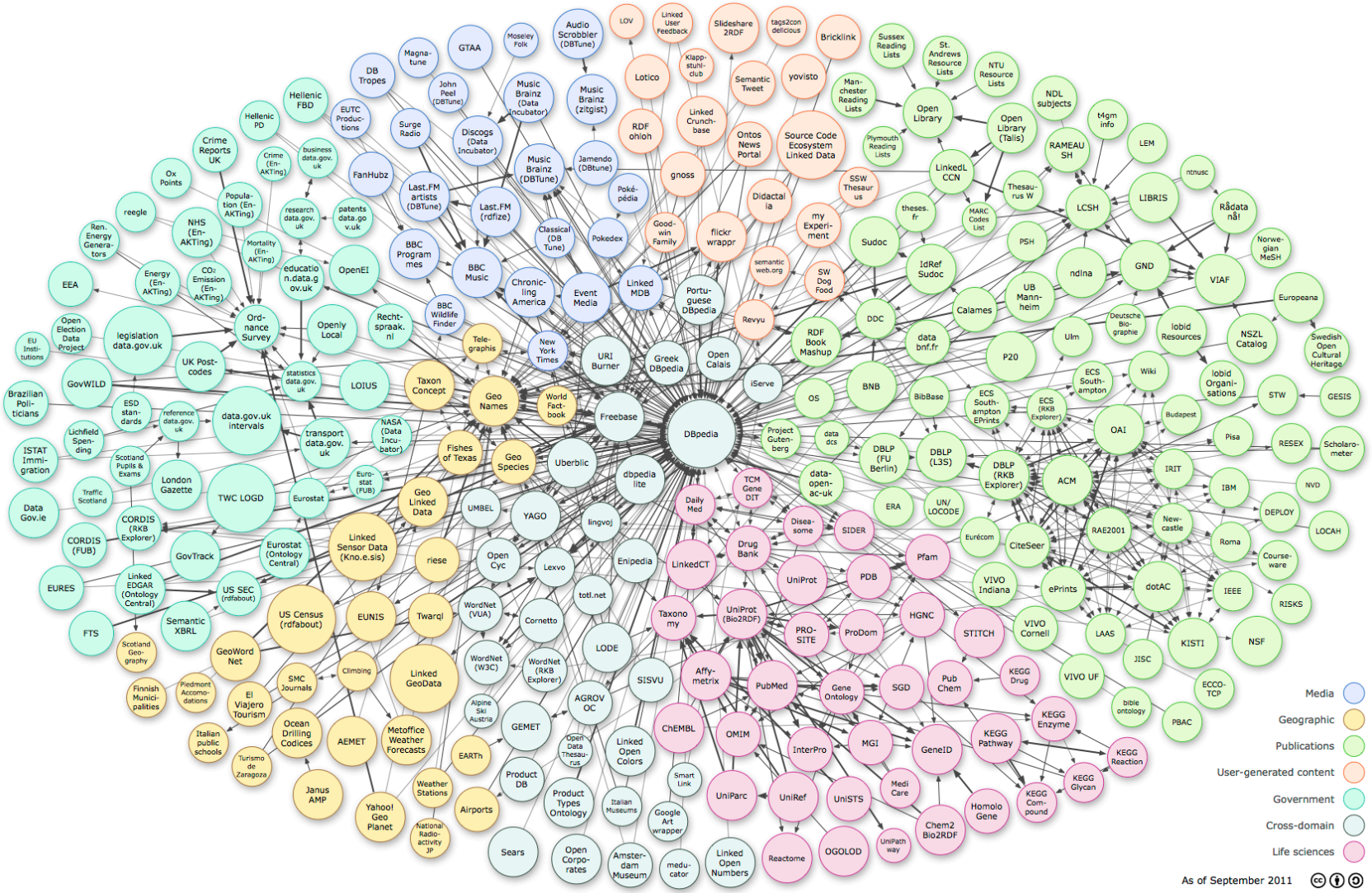
2.4. SEGURIDAD Y BIG DATA

La materia prima de los procesos y tecnologías de Big Data es la información, el dato. Desde el punto de vista legal, nos surge una pregunta, ¿Qué podemos hacer con ellos? Para responderla se deben tener en cuenta sobre todo cual es la legislación vigente, en España, por ejemplo hablamos de LOPD (Ley Orgánica de Protección de Datos) que es una de las más restrictivas del mundo. Otro ejemplo podrían ser los robustos protocolos implantados para la solicitud de datos a las compañías telefónicas, que para mayor seguridad, están auditados.

Por otro lado, para hacer respetar estas leyes y protocolos de protección de datos, España tiene grupos en las Fuerzas y Cuerpos de Seguridad del Estado con un reconocido prestigio.

Otra forma de mantener la seguridad y privacidad de los datos que se analizan es trabajar siempre con datos agregados y anonimizados. Sin embargo, pueden existir casos problemáticos, en los que aún teniendo el dato anonimizado, al cruzar la información con otras fuentes de información se puede descubrir el propietario de los datos.

Hoy en día existen muchas fuentes de información pública, denominadas Open Data. En el siguiente gráfico se muestran muchas de ellas.


DBpedia
Freebase
Geo Names
YAGO
UMBEL
Uberblic
dbpedia lite
Enipedia
lingvoj
totl.net
LODE
Lexvo
Open Cyc
riese
Linked Sensor Data (Kno.e.sis)
Linked GeoData
EUNIS
Twarql
WordNet (VUA)
WordNet (W3C)
WordNet (RKB Explorer)
Cornetto
SISVU
AGROVOC
GEMET
Product DB
Product Types Ontology
Linked Open Colors
Open Data Thesaurus
Smart Link
Google Art wrapper
Italian Museums
Amsterdam Museum
medu-cator
Linked Open Numbers
Reactome
OGOLOD
UniPath way
Chem2 Bio2RDF
Homolo Gene
KEGG Com-pound
KEGG Glycan
KEGG Reaction
KEGG Enzyme
KEGG Pathway
KEGG Drug
GeneID
Medi Care
UniSTS
UniRef
UniParc
InterPro
MGI
OMIM
ChEMBL
Affy-metrix
PubMed
Gene Ontology
SGD
Pub Chem
Taxono my
UniProt (Bio2RDF)
PRO-SITE
ProDom
HGNC
STITCH
UniProt
PDB
Pfam
LinkedCT
Drug Bank
Disea-some
SIDER
TCM Gene DIT
Daily Med
data-open-ac-uk
ERA
UN/ LOCODE
Eurécom
CiteSeer
VIVO Indiana
VIVO Cornell
ePrints
dotAC
RAE2001
New-castle
ACM
IRIT
IBM
NVD
DEPLOY
LOCAH
Roma
Course-ware
IEEE
RISKS
LAAS
JISC
KISTI
NSF
VIVO UF
ECCO-TCP
bible ontology
PBAC
DBLP (RKB Explorer)
DBLP (L3S)
DBLP (FU Berlin)
data dcs
Project Guten-berg
OS
iServe
Greek DBpedia
Open Calais
URI Burner
Portu-guese DBpedia
Linked MDB
Event Media
Chronic-ling America
BBC Music
BBC Programmes
BBC Wildlife Finder
Tele-graphis
New York Times
Revyu
RDF Book Mashup
Taxon Concept
Geo Species
Fishes of Texas
NASA (Data Incu-bator)
Euro-stat (FUB)
Eurostat
Eurostat (Ontology Central)
US Census (rdfabout)
US SEC (rdfabout)
Linked EDGAR (Ontology Central)
Semantic XBRL
GeoWord Net
SMC Journals
El Viajero Tourism
Ocean Drilling Codices
Italian public schools
Piedmont Accomo-dations
Finnish Munici-palities
Scotland Geo-graphy
FTS
EURES
CORDIS (FUB)
CORDIS (RKB Explorer)
GovTrack
Climbing
AEMET
Metoffice Weather Forecasts
EARTh
Weather Stations
National Radio-activity JP
Airports
Janus AMP
Yahoo! Geo Planet
Sears
Open Corpo-rates
Turismo de Zaragoza
Traffic Scotland
Lichfield Spen-ding
ISTAT Immi-gration
Data Gov.ie
Brazilian Poli-ticians
GovWILD
EEA
EU Insti-tutions
Open Election Data Project
legislation data.gov.uk
UK Post-codes
data.gov.uk intervals
London Gazette
TWC LOGD
ESD stan-dards
reference.data.gov.uk
Scotland Pupils & Exams
statistics.data.gov.uk
LOIUS
transport data.gov.uk
Ord-nance Survey
Openly Local
Recht-spraak.nl
education.data.gov.uk
OpenEI
Ren. Energy Genera-tors
Energy (En-AKTing)
CO2 Emission (En-AKTing)
Mortality (En-AKTing)
NHS (En-AKTing)
Popula-tion (En-AKTing)
Crime (En-AKTing)
business.data.gov.uk
research.data.gov.uk
patents.data.gov.uk
reegle
Ox Points
Crime Reports UK
Hellenic PD
Hellenic FBD
EUTC Produc-tions
DB Tropes
Magna-tune
Moseley Folk
GTAA
John Peel (DBTune)
Surge Radio
FanHubz
Discogs (Data Incubator)
Last.FM artists (DBTune)
Last.FM (rdfize)
Music Brainz (Data Incubator)
Music Brainz (zitgist)
Audio Scrobbler (DBTune)
Jamendo (DBTune)
Classical (DB Tune)
Pokedex
Poké-pédia
gnoss
RDF ohloh
Lotico
LOV
Linked User Feedback
Klapp-stuhl-club
Linked Crunch-base
Semantic Tweet
Slideshare 2RDF
tags2con delicious
yovisto
Bricklink
Source Code Ecosystem Linked Data
Ontos News Portal
Didactalia
Good-win Family
flickr wrappr
my Experi-ment
SW Dog Food
semantic web.org
Sussex Reading Lists
St. Andrews Resource Lists
NTU Resource Lists
Man-chester Reading Lists
Plymouth Reading Lists
SSW Thesaurus
Open Library
Open Library (Talis)
LinkedL CCN
NDL subjects
t4gm info
LEM
RAMEAU SH
Thesau-rus W
MARC Codes List
theses.fr
LCSH
LIBRIS
ntnusc
Råda ta nå!
Norwe-gian MeSH
PSH
Sudoc
IdRef Sudoc
ndlna
GND
VIAF
Europeana
Swedish Open Cultural Heritage
UB Mann-heim
Calames
DDC
data bnf.fr
Ulm
Deutsche Bio-graphie
lobid Resources
NSZL Catalog
lobid Organi-sations
P20
BNB
BibBase
ECS South-ampton
ECS South-ampton EPrints
ECS (RKB Explorer)
Wiki
Budapest
OAI
Pisa
RESEX
Scholaro-meter
STW
GESIS
Media
Geographic
Publications
User-generated content
Government
Cross-domain
Life sciences
As of September 2011

Otro caso típico de este tipo de vacío o incertidumbre legal es aquel en el que una empresa ofrece servicios en España, pero tiene los servidores en otro país con una normativa de protección de datos diferente, ¿Qué legislación prevalece? En teoría, en este caso debería prevalecer la española que es donde presta los servicios la empresa, sin embargo, al no tener los servidores dentro de España, es muy difícil de comprobar y verificar el cumplimiento de la ley.

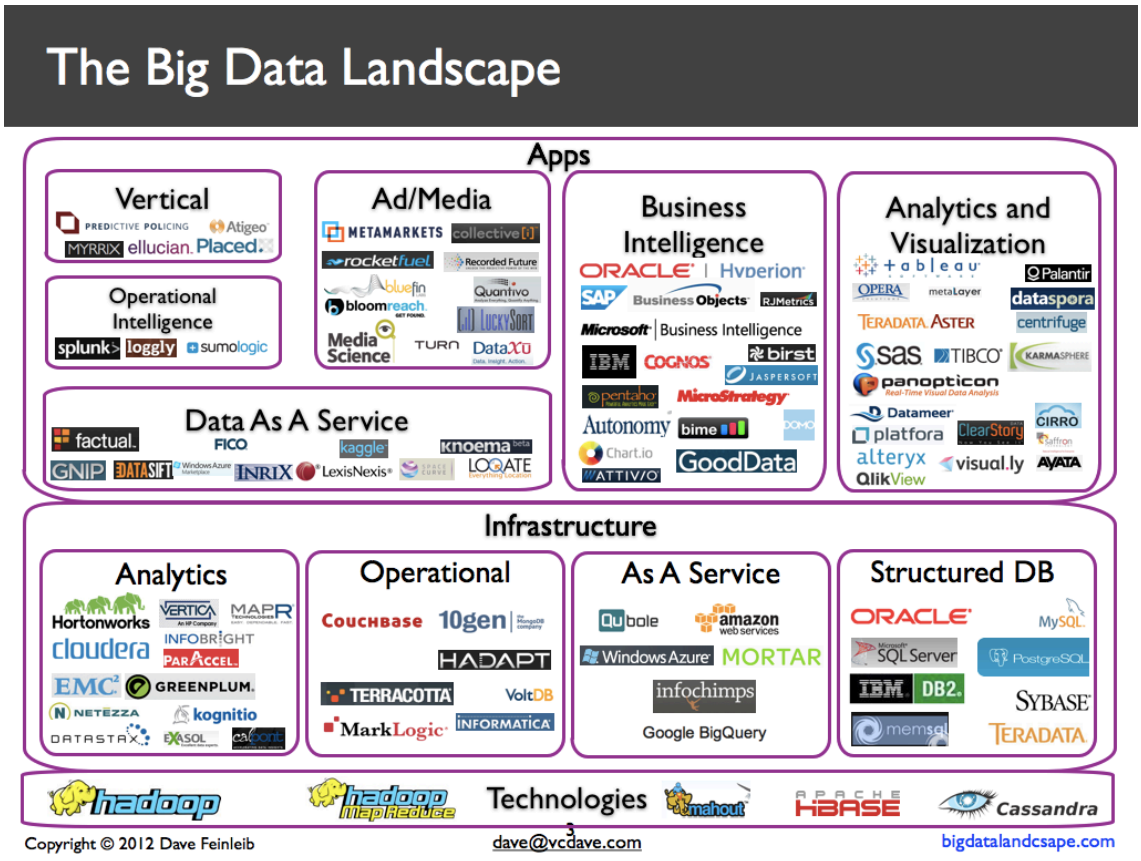
Una solución que parece que se está aceptando mayoritariamente en el mercado es acudir a empresas específicas que se encargan de certificar que el tratamiento de los datos que se está realizando es el adecuado. Esta solución se basa en el prestigio que estas empresas específicas tengan, esto es esencial para dotar de confianza al proceso.

3. BIG DATA LANDSCAPE.

3.1. INTRODUCCIÓN

En este apartado se va describir de forma sencilla uno de los landscape típicos que se pueden encontrar habitualmente cuando se realiza una búsqueda en internet.

Como base para este punto se va a usar el siguiente landscape



En él se puede observar una clasificación principal en base a tecnologías, infraestructura y aplicaciones, y dentro de cada una de ellas, varias subcategorias:

- ✓ Apps.
  - o Vertical
  - o Operational Intelligence
  - o Business Intelligence
  - o Analytics and Visualization

- Ad/Media Apps
- Data as Service

✓ Infrastructure.

- Analytics
- Operational
- As a Service
- Structured DB

✓ Technologies

- Hadoop
- Mahout
- HBASE
- Cassandra

A continuación se va a detallar este último punto, el de las tecnologías usadas en el mundo del Big Data, para poder comprender un poco mejor como funcionan.

3.2. TECNOLOGÍAS.

En la actualidad existen principalmente cuatro tecnologías usadas para el análisis de big data. Estas tecnologías son:

- ✓ Apache HBASE
- ✓ Cassandra
- ✓ Apache Hadoop MapReduce
- ✓ Mahout

3.2.1. Apache Hadoop Mapreduce.

Hadoop fue creado por Doug Cutting en el año 2006. Su nombre viene de un elefante de juguee que tenía cuando era niño. Su objetivo originalmente era ayudar en un proyecto de motor de búsqueda, denominado Nutch.

Apache Hadoop tiene licencia libre y es software usado en aplicaciones distribuidas. Una de las ventajas que tiene es que puede llegar a gestionar aplicaciones con miles de nodos y enormes cantidades de datos.

La aparición de Hadoop fue posible gracias al estudio de dos documentos de Google, el paper sobre MapReduce de 2004 y el paper de Google File System (GFS) de 2003.

Hadoop es un proyecto de Apache que está en constante evolución gracias a la gran comunidad de usuarios y contribuyentes entre los cuales destacan twitter, linkedIn, Facebook, etc…. Esta hecho en el lenguaje de programación Java. Yahoo! ha sido el mayor contribuyente de este proyecto.

3.2.1.1. Arquitectura.

El principal componente de la arquitectura de hadoop es Hadoop common. Esta pieza es laque tiene las librerías y funciones comunes que dan soporte al resto de modulos de Hadoop. Al ser software libre, al obtenerlo, además de los ficheros .jas con los ejecutables, vienen los scripts con los que se lanza Hadoop. Por otro lado también se proporciona la documentación del software y el código fuente.

La funcionalidad básica de Hadoop se basa en la existencia de un nodo maestro que contiene un gestor de trabajos, un gestor de tareas y además conoce los metadatos (índice de nombres) que indican la dirección exacta de cada archivo. A parte, puede contener los propios datos, aunque no es lo habitual. Y por otro lado múltiples nodos de datos, que aparte de los datos también tienen gestor de tareas. Esto es así para que se puedan ejecutar las tareas también en los propios servidores de datos, reduciendo notablemente el tráfico en la red.

Para poder proporcionar una alta tolerancia a fallos, se usa la redundancia o replicación, es decir, un dato se copia en al menos tres servidores diferentes. De esta forma si uno de los servidores falla, ya sea por problemas de energía o por algún otro error, el dato siempre estará disponible en los otros dos servidores. También tiene la ventaja de que si al acceder a un servidor a por el dato, el servidor está ocupado, se puede enviar el trabajo a otro servidor en el que este copiado este dato, evitando esperas y por lo tanto haciendo que el sistema sea más eficiente.

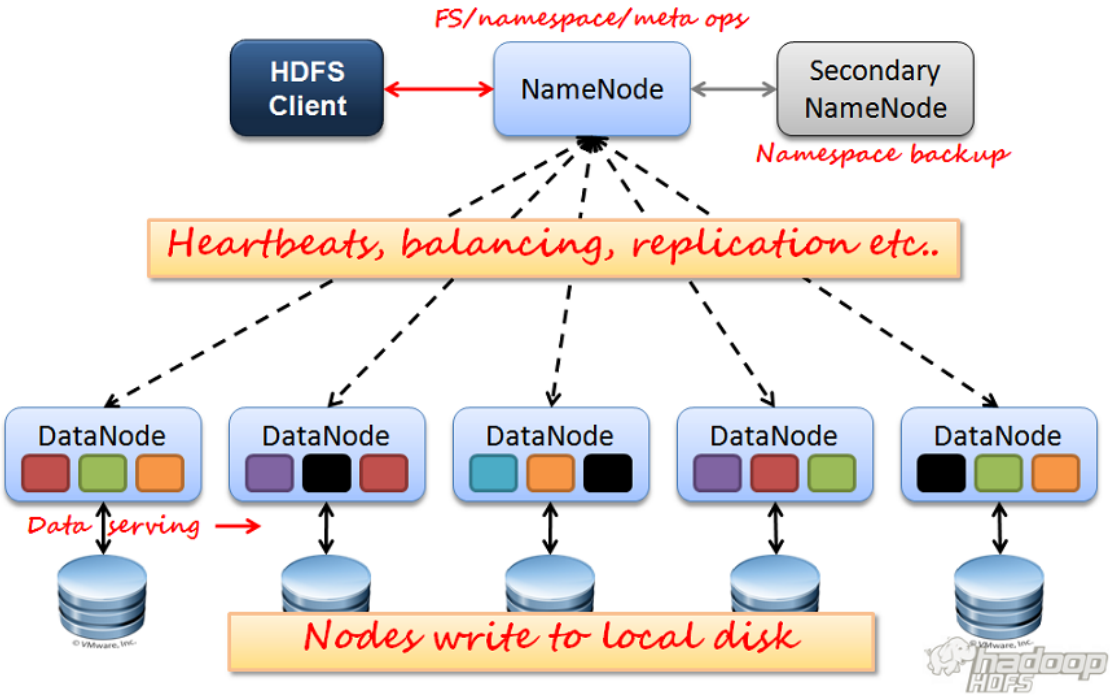
Un requisito mínimo de Hadoop es tener instalado en el clúster JRE 1.6 o superior, y SSH.

3.2.1.2. Sistema de Archivos (HDFS)

El Hadoop Distributed File System (HDFS) es el sistema de almacenamiento de archivos distribuido que usan las aplicaciones de Hadoop.

Este sistema está desarrollado en Java para el framework Hadoop y sus características principales son la eficiencia (con ficheros grandes) y la tolerancia a fallos.

A continuación se puede ver un sistema HDFS típico que se monta sobre Hadoop, con un único servidor maestro (a veces tiene un secundario como back up) y varios servidores de datos.



Los protocolos usados en este sistema son:

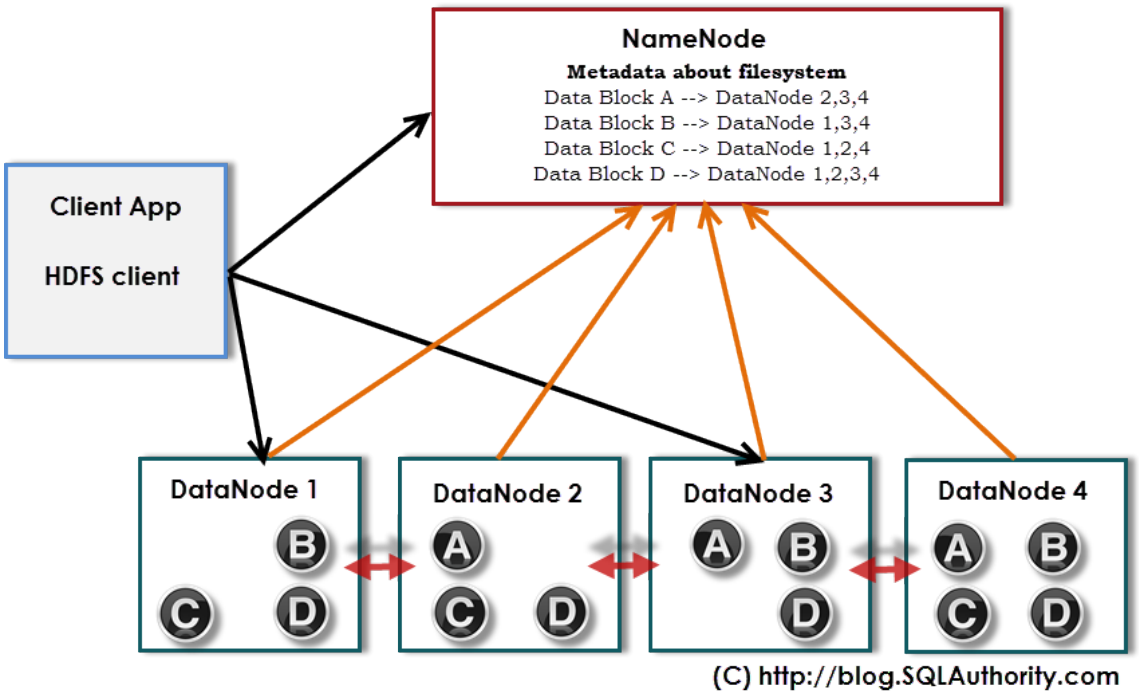
- TCP/IP para la comunicación entre los servidores y la red.
- RPC, Remote Procedure Call para la comunicación entre los clientes.

El HDFS es eficiente almacenando archivos grandes (el tamaño ideal de archivo es de 64 MB). Tal y como se ha mencionado anteriormente consigue tener una alta tolerancia a fallos gracias al replicado de datos a través de varios servidores de datos.

HDFS no necesita almacenamiento RAID, aunque para ciertas configuraciones sigue usándose para aumentar el rendimiento.

En general, el valor por defecto de replicación en estos sistemas es tres. Además las copias suelen estar dos en el mismo servidor de datos y otra en otro servidor diferente.

Otra funcionalidad destacada de HDFS es que los servidores de datos tienen la capacidad de comunicarse entre ellos de cara a balancear la carga de datos entre todos ellos.



Además de la alta tolerancia a fallos, lo que se busca con esta manera de distribuir los datos es poder usar como servidores en estos sistemas, hardware de bajo coste. Máquinas como las que cualquier persona podría tener en su casa. Esto es así porque aunque este hardware tiene mayor probabilidad de fallar, el error de una máquina no afecta al sistema completo. Esto implica que este tipo de sistemas se pueden montar de forma mucho más barata que otras soluciones que existen en el mercado, dato que las empresas de hoy en día tienen muy en cuenta.

En algunas configuraciones de HDFS se incluye también un servidor maestro secundario. Este servidor se usa como backup para el caso en el que el servidor maestro original falla. Este servidor secundario se conecta periódicamente con el servidor maestro principal para almacenar copias de la información. Estas copias se podría usar después para reiniciar el servidor primario, por ejemplo.

Hadoop podría trabajar con otros sistemas de archivos que no fueran HDFS, sin embargo, si no se usa su propio sistema de archivos se pierde una de sus principales características, la localidad y ya no podría reducir el tráfico de red, ya que no sabría que servidor es el más próximo a los datos.

HDFS fue diseñado para gestionar archivos muy grandes. Con archivos de tamaño pequeño no es eficiente. Otra característica es que no proporciona alta disponibilidad, es decir, la latencia es alta.

3.2.1.3. Mapreduce

MapReduce es un modelo de programación, y una aplicación asociada a la transformación y la generación de grandes conjuntos de datos.

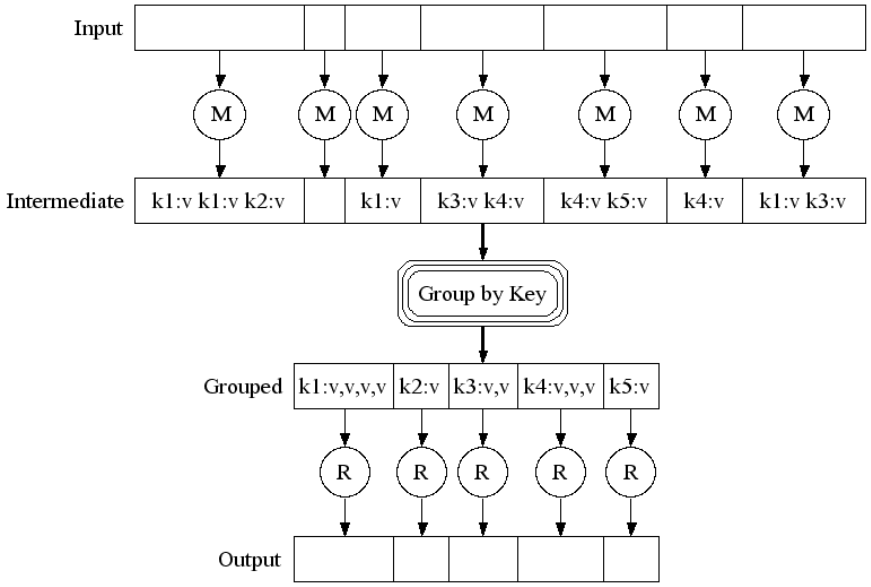
El nombre procede de sus dos principales métodos o funciones, map, que se encarga convertir el problema en una serie de pares clave/valor, y reduce que se encarga de combinar todos estos valores intermedios asociados a la misma clave. Muchas tareas del mundo real son expresables de esta forma, aunque no todas y por lo tanto, Mapreduce no sirve para resolver cualquier problema, sino solo aquellos que puedan expresarse en pares clave/valor.

Los programas que se escriben en este estilo pueden ser paralelizados automáticamente y ejecutados dentro de un cluster con múltiples maquinas de bajo coste. El propio sistema, en tiempo de ejecución, se preocupa de mapear los datos de entrada y partirlos a lo largo de varias máquinas, controlando cualquier error en alguna de ellas (la tareas que estuviera realizando se la asignaría a otra máquina), y controlando también la comunicación entre estas máquinas. Esto permite que los programadores no tengan que preocuparse de gestionar los recursos paralelizando, porque ya lo hace el sistema de forma automática.

Este tipo de sistemas por lo tanto tienen como una de sus principales características, la tolerancia a fallos. El servidor maestro, para detectar si un nodo ha fallado, envía periódicamente heartbeats al resto de nodos. Al detectar un fallo, dependiendo de que tar´´ea estuviera realizando se volvería a ejecutar entera o solo una parte. Si estaba realizando un Map(), se volvería a ejecutar entera, si era Reduce() lo que estaba ejecutándose, solo se re-ejecutaría esta parte. El servidor maestro tiene el control de cada tarea porque todos los nodos le comunican al servidor maestro su finalización. En el caso de que fallase el servidor maestro, las primeras implementaciones, incluso algunas hoy en día, no tienen en cuenta este supuesto por ser poco probable, sin embargo, muchas de ellas ya lo resuelven a través de un servidor maestro secundario, en el que se van realizando copias periódicas del primario y en caso de fallo entraría en funcionamiento.

Por otro lado, usan también la redundancia de cara a optimizar el sistema, evitando problemas no solo de errores en servidores, sino también los problemas de lentitud en ciertos servidores.

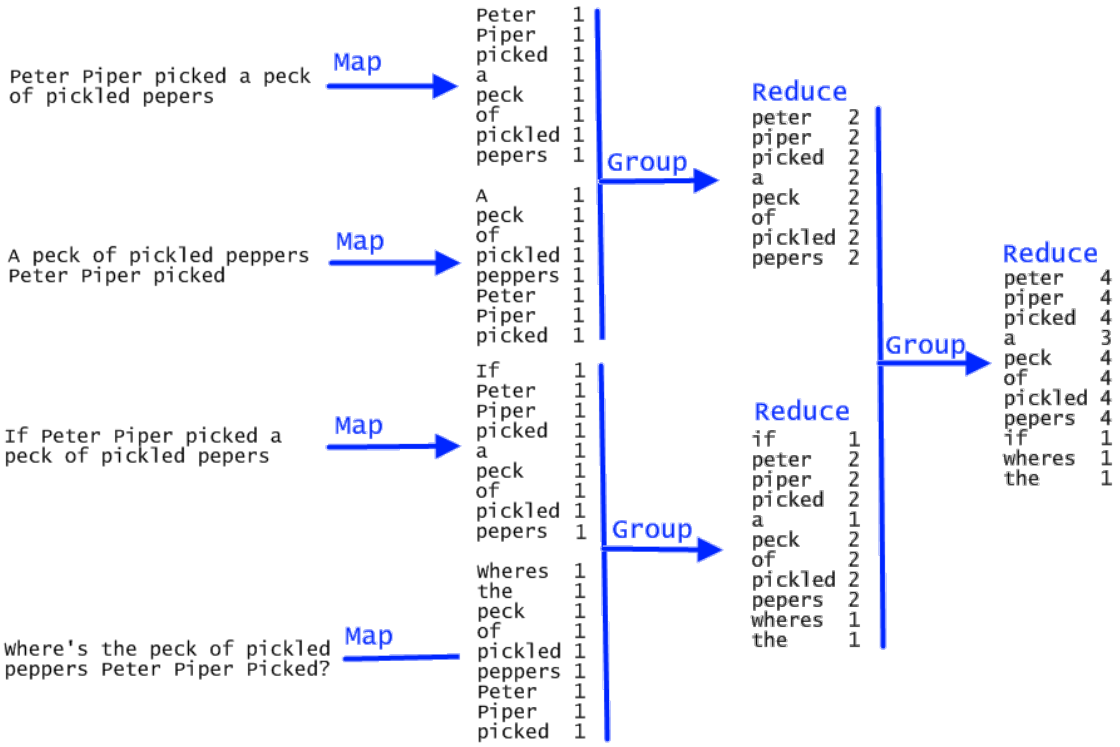
Por último, mapreduce también tiene una función de monitorización de tareas y de control de estado.



Es esta imagen se puede ver un ejemplo esquemático de cómo sería la ejecución de mapreduce para resolver un problema a través del uso de Map() y Reduce().

La función Map() toma los datos de entrada y produce un conjunto de pares clave/valor. Una librería estándar de mapreduce se encarga de agrupar todos los pares con la misma clave y pasárselos a la función Reduce(). Esta función toma como entrada con una clave y un conjunto de valores y forma una nueva clave intermedia fusionando estos valores.

A continuación se muestra un ejemplo típico de de uso de esta función mapreduce en el que a partir de varios textos se cuentan el número de apariciones de cada una de las palabras contenidas en ellos.



Como se puede observar en la imagen, la solución del problema es muy sencilla usando las funciones map() y reduce().

Un framework mapreduce además de en estas dos funciones, se basa en lo que se puede llamar coloquialmente “divide y vencerás”. Cada una de las tareas individuales de map() o de reduce() se podría ejecutar en un nodo diferente paralelizando su ejecución con el consiguiente beneficio en la eficiencia.

Aunque en este ejemplo no parezca tener importancia, un framework mapreduce suele trabajar con petabytes de datos y para procesar tanta información, no solo es importante la eficacia, sino que también lo es la eficiencia. Para conseguirla, el framework usa, tal y como hemos visto en los párrafos anteriores, un nodo maestro que divide el problema principal en muchas subtareas, y cada una de estas subtareas se las encarga a uno de los muchos nodos de trabajo que suele tener el framework mapreduce, ejecutándose de esta forma muchas tareas de forma paralela. Una vez los nodos de trabajo van terminando la tarea que le ha solicitado el nodo maestro, estos le devuelve el resultado al nodo maestro.

Aunque se habla de paralelizar subtareas, no todas las subtareas pueden ejecutarse en paralelo, sino que algunas dependen de la finalización de otras. Por ejemplo, en la imagen anterior, las dos funciones map() ejecutadas sobre los dos primeros textos, deben ejecutarse

antes que la función reduce() que está a su derecha. Esto se debe a que la función reduce() tiene que usar el resultado de las dos funciones map() previas como input.

El control de que todas las subtareas se ejecuten en el orden correcto y que los tiempos de espera sean los mínimos necesarios se realiza en el nodo maestro.

Los pasos que sigue por lo tanto el framework mapreduce son cinco:

- ✓ Preparar los input de las funciones map().
- ✓ Ejecutar las funciones map() correspondientes.
- ✓ Mezclar la salida de estas funciones, preparándolas como input para las funciones reduce().
- ✓ Ejecutar las funciones reduce() correspondientes().
- ✓ Producir la salida o resultado correspondiente.

Simplificando mucho, se podría decir que Mapreduce es el equivalente a un SELECT y un GROUP BY de una base de datos relacional, cuando la base de datos relacional es muy grande.

3.2.2. Cassandra.

Apache Cassandra es un motor de base de datos NoSQL, open source desarrollado en Java y creada originalmente por facebook, que fue donada en 2009 a Apache como software libre.

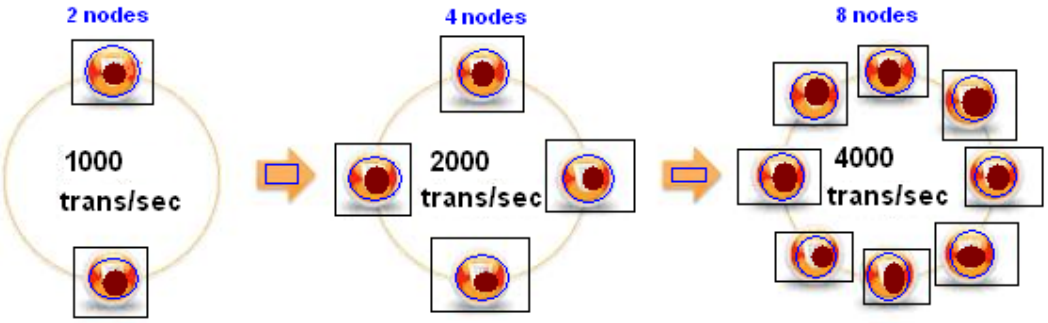
Cassandra está diseñada especialmente para manejar enormes cantidades de datos (hasta varios Terabytes) a lo largo de varios servidores y en la nube. Proporciona alta disponibilidad, escalabilidad lineal y una gran simplicidad operacional al ser instalado en commodity software que no tiene un único punto de error (es decir, usa la replicación de datos). Además su modelo de datos está diseñado para proporcionar una flexibilidad máxima y unos tiempos de respuesta muy rápidos.

¿En que se basa Cassandra?

- ✓ En una arquitectura de nodos en la que no existe un nodo maestro, sino que todos los nodos son iguales. Es decir, ni el programador de la base de datos ni su administrador tienen que preocuparse de la distribución de la información entre los diferentes nodos

que compongan la base de datos, sino que ella misma se encarga de distribuirlos a lo largo de todos los nodos que pertenecen a un "ring" o cluster de la base de datos.

- ✓ En la replicación de datos, almacenando copias de los datos en los nodos que pertenecen al ring de cassandra. Esto significa que si algún nodo de la base de datos falla, una o más copias de esos datos estarán disponibles en otros nodos del cluster. El número de copias que se realizan de los datos se puede configurar. Otra característica que se puede configurar es adaptar la replicación para trabajar en un centro de datos, en varios centros de datos o en múltiples zonas disponibles en la nube.
- ✓ En la escalabilidad lineal, esto quiere decir que se puede aumentar fácilmente la capacidad, simplemente aumentando el número de nodos disponibles de la base de datos.



Cassandar ring showing sclability

3.2.2.1. Cassandra Query Language (CQL).

El lenguaje por defecto o interfaz principal de Cassandra DBMS es el Cassandra Query Language (CQL).

Usar CQL es muy parecido a usar SQL (Structure Query Language). La principal diferencia es que CQL no soporta JOINS de tablas ni subqueries, excepto para análisis batch a través de HIVE (software de almacenamiento de datos de Apache que facilita la consulta y gestión de grandes conjuntos de datos que residen en almacenamiento distribuido como por ejemplo Cassandra o HDFS).

La simplicidad para leer y usar CQL es una de las ventajas de las nuevas versiones de Cassandra con respecto a las anteriores que usaban API´s complejas para realizar el mismo trabajo.

3.2.2.2. Arquitectura de Cassandra.

Para describir la arquitectura usada por Cassandra, primero debemos conocer varios términos que se van a usar en los próximos párrafos con este fin. Estos términos son los siguientes:

- ✓ Cluster.

  Es un conjunto de máquinas que dan soporte a Cassandra, en las cuales se almacenan los datos, y que son vistas por los clientes de la base de datos como una única máquina. Un cluster contiene uno o más Data Centers.

- ✓ Data Center.

  Es un grupo de nodos relacionados entre sí en un cluster y configurados para proporcionar replicación y reparto de carga de trabajo. No necesariamente tiene que ser un data center físico, sino que pueden configurarse Data Centers lógicos. Dependiendo del factor de replicación, los datos pueden ser copiados en uno o en más de un data center.

- ✓ Commit Log.

  Todos los datos son escritos primero en el commit log para asegurar la durabilidad de los datos. Después de que todos los datos hayan sido descargados a las SSTables, los commit log pueden ser archivados, borrados o reutilizados.

- ✓ SSTable.

  Sort String Table (SSTable) es un fichero de datos que no puede ser modificado y en el cual cassandra escribe memtables periódicamente. Solo se pueden añadir datos a las SSTable, nunca borrar o modificar. Se almacenan en el disco de forma secuencial. Una SSTable se mantiene para cada Table o Column Family.

- ✓ Table o Column Family.

  Es lo que entenderíamos como tabla en un modelo de base de datos relacional. Se trata de un contenedor de un conjunto ordenado de columns. Cada Column Family se almacena en un archivo separado.

- ✓ Column.

  Es la unidad básica en el modelo de datos de Cassandra. Una columna es un triplete compuesto por Key o nombre, Value y Timestamp. Todas las columnas son inmutables, es decir, no se pueden modificar, solo crear y eliminar. La razón para que no puedan alterarse una columna es evitar problemas con el multithreading. Las columnas están organizadas dentro de Columns Families.

- ✓ Super Column.

  Es una Column cuyos valores son una o más columns, que en este contexto se las denominará SubColumns. El número de SubColumns que se pueden definir en una Super Column es ilimitado. Las Super Columns a diferencia de las Column, no tienen un timestamp definido.

La arquitectura de Cassandra está basada en el entendimiento de que los fallos de sistema y de hardware pueden ocurrir y de hecho ocurren. Cassandra solventa el problema de los fallos empleando un sistema distribuido peer-to-peer (P2P) a lo largo de nodos homogéneos en los que los datos están distribuidos por todos los nodos del cluster.

Cada nodo intercambia información con el cluster cada segundo. Existe un log secuencial de confirmaciones de escritura en cada nodo que captura toda la actividad de escritura en la base de datos para asegurar la durabilidad de los datos. Los datos son entonces indexados y escritos en una estructura de memoria que se parece a una writeback cache. Una vez que la memoria de esta "cache" está llena, se vuelcan los datos a disco en un fichero de datos SSTable. Todo lo que se escribe es automáticamente particionado y replicado a lo largo del clúster. Cassandra, empleando un método llamado "Compaction", periódicamente consolida las SSTables, descartando datos obsoletos y "tombstones" (datos que han sido marcados indicando que son datos borrados).

La arquitectura de Cassandra permite que cualquier usuario autorizado se conecte a cualquier nodo de cualquier data center y acceda a sus datos usando el CQL. Para que sea más fácil su uso, CQL tiene una sintaxis muy parecida a SQL. Desde el punto de vista de CQL, la base de datos consiste en tablas.

Por norma general, un cluster suele tener un keyspace por aplicación.

Las peticiones de lectura o escritura de un cliente pueden ser enviadas a cualquier nodo dentro del cluster. Cuando un cliente se conecta a un nodo para hacer una petición de lectura o escritura, ese nodo sirve de coordinador para esa operación en particular. El coordinador actúa como un proxy entre el cliente y el resto de nodos en los que están los datos que necesita para su petición. El coordinador es el que decide qué nodos del ring deben atender la petición basándose en la configuración del cluster.

Cassandra Write Data Flows

Single Region, Multiple Availability Zone

1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk

If a node goes offline, hinted handoff completes the write when the node comes back up.

Requests can choose to wait for one node, a quorum, or all nodes to ack the write

SSTable disk writes and compactions occur asynchronously

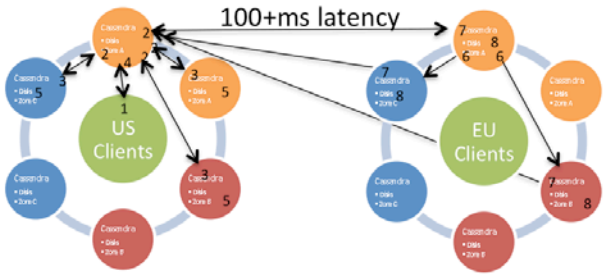
NETFLIX

Data Flows for Multi-Region Writes

Consistency Level = Local Quorum

1. Client Writes to any Cassandra Node
2. Coordinator node replicates to other nodes Zones and regions
3. Local write acks returned to coordinator
4. Client gets ack when 2 of 3 local nodes are committed
5. Data written to internal commit log disks
6. When data arrives, remote node replicates data
7. Ack direct to source region coordinator
8. Remote copies written to commit log disks

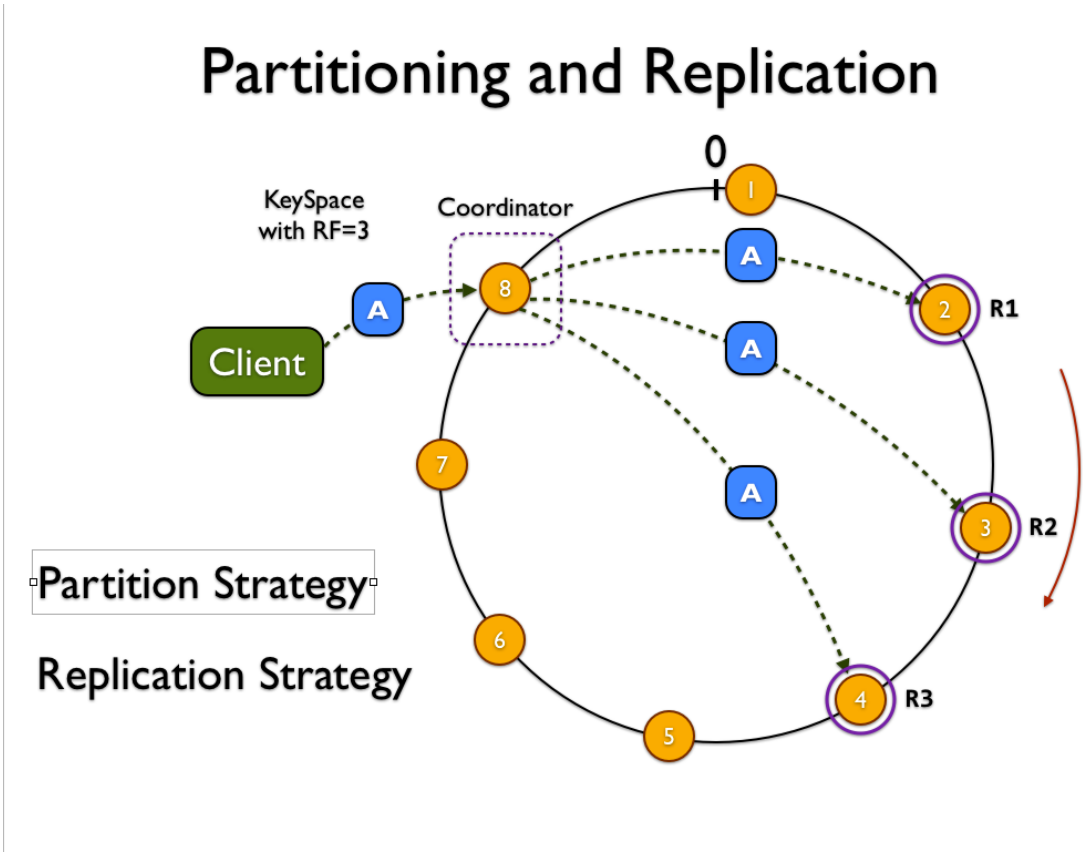
If a node or region goes offline, hinted handoff completes the write when the node comes back up. Nightly global compare and repair jobs ensure everything stays consistent.



NETFLIX

Los componentes clave para la configuración de cassandra son:

- ✓ Gossip. Es un protocolo de comunicación peer to peer para descubrir y compartir información sobre la localización y el estado de otros nodos en un cluster de Cassandra. La información de Gossip también se conserva localmente en cada nodo para que pueda ser usada inmediatamente cuando un nodo se reinicia.
- ✓ Partitioner. Es el encargado de determinar cómo se distribuyen los datos a lo largo de los nodos en un cluster y en que nodo almacenar la primera copia de los datos. Básicamente es una función hash para el cálculo del token de una "partition key". Cada fila de datos es identificada unívocamente por una partition key y distribuida a lo largo del cluster por el valor del token. En la configuración del partitioner se decide el número de tokens de un nodo. El número de tokens que le asignes a un nodo, va a depender de las capacidades del hardware. Si no van a existir nodos virtuales, se debe usar la configuración initial_token.
- ✓ Replication Factor. Es el número total de copias de un dato que se van a almacenar a lo largo del cluster. Un replication factor de uno, significa que solo existirá una copia de cada fila. Un replication factor de dos, implica que existirán dos copias, una en cada nodo. Cada replica es igual de importante, no existe una réplica principal y otra secundaria. El replication factor se define a nivel de data center. Por norma general, el replication factor debe ser mayor que uno y menos que el número total de nodos de un cluster.
- ✓ Replica Placement Strategy. Cassandra almacena copias o replicas de los datos en múltiples nodos para asegurar la fiabilidad y la tolerancia a fallos. Una estrategia de replicación decidirá en que nodos se almacenarán las diferentes replicas de los datos. La primera réplica es simplemente la primera copia, no es única en ningún sentido. La estrategia comúnmente más utilizada es la NetworkTopologyStrategy porque es la que mejor se expande a multiples datacenter cuando se necesita una expansión de la base de datos. Cuando creas el keyspace, la estrategia de replicado es una de las características que se debe definir.



3.2.3. Apache HBase.

Apache HBase es una base de datos distribuida, desarrollada en java, en código abierto y que se basa principalmente en el artículo que Google sacó a finales de 2006 sobre BigTable (Un sistema de almacenamiento distribuido de datos estructurados que soporta claves de búsqueda: búsqueda por rango y análisis de archivo de alto rendimiento). BigTable es la tecnología que permite que Google escale toda su infraestructura a cientos de servidores de bajo coste de una manera homogénea y sin depender de las restricciones de una base de datos relacional como MySQL, Oracle o cualquier otra.

Un año después de este artículo de Google, aparece la primera versión de HBase funcionando sobre Hadoop.

HBase proporciona a HDFS las mismas características que ofrece BigTable a Google File System. Estas características se pueden resumir en:

- ✓ Escalabilidad lineal y modular
- ✓ Consistencia estricta en lecturas y escrituras de datos.
- ✓ Sharding automático y configurable en las tablas.

- ✓ Gestión y tolerancia a fallos entre regiones.
- ✓ Tablas convenientemente adaptadas para servir a los trabajos de Hadoop Mapreduce.
- ✓ API hecha en java fácil de usar para permitir el acceso de los diferentes clientes a la base de datos.
- ✓ Bloqueo de caché y "bloom" de filtros para consultas en tiempo real.

El objetivo principal del proyecto era tener la capacidad de poder almacenar tablas muy grandes (con billones de filas por varios millones de columnas), cumpliendo con todas las características mencionadas anteriormente (sobre todo la última que habla de tener tiempos de acceso de lectura/escritura muy bajos), todo ello en un cluster compuesto por hardware básico de bajo coste.

El proyecto de HBase, al ser de código abierto, ha tenido muchos contribuyentes importantes. Uno de ellos es por ejemplo Facebook, que lo empezó a usar en 2010 para su sistema de mensajería.

¿Por qué HBase si Hadoop Distribute File System ya sirve por si solo para almacenar grandes cantidades de información? ¿Cuál es la diferencia entre HBase y Hadoop? Estas son preguntas que surgen frecuentemente cuando una persona comienza a indagar en este mundo.

HDFS es un sistema distribuido de archivos que funciona muy bien para almacenar grandes archivos, pero no es un sistema de archivos al uso y no proporciona un acceso rápido a un archivo individual. HBase, por el contrario, se monta encima de HDFS y proporciona búsquedas de registros rápidas (y actualizaciones) en una gran tabla.

HBase internamente indexa los datos que existen en HDFS para poder realizar con ello después búsquedas muy rápidas.

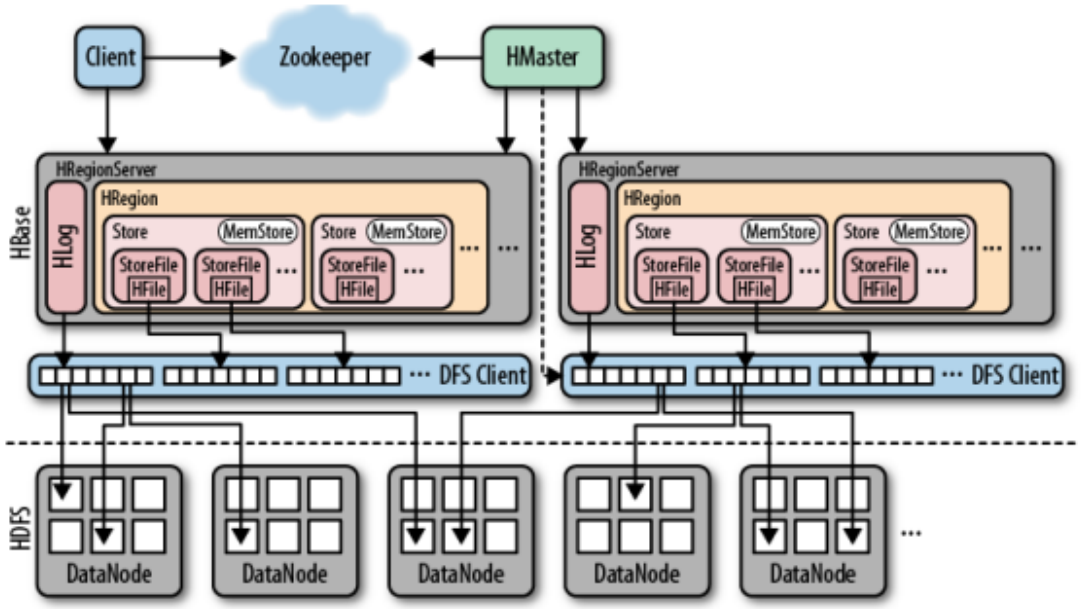
3.2.3.1. Arquitectura de HBase.

La arquitectura de HBase se basa en dos tipos de servidores diferentes:

- ✓ Maestros (HBase Master)
- ✓ Esclavos (HBase Region)

Cada uno de ellos necesita una configuración y unas especificaciones diferentes. Por ejemplo, el servidor maestro no necesita tanto espacio como un servidor esclavo, por lo que no hace falta que un servidor maestro tenga mucha memoria.

A continuación podemos ver un cluster de HBase montado sobre Hadoop.



En esta figura se puede observar como HBase utiliza dos tipos de archivos diferentes (HLog y HFile), el primero de ellos se utiliza para el registro de escritura (Write Ahead Log o WAL), y el otro tipo es para el almacenamiento de datos reales.

Los archivos son gestionados principalmente por los servidores esclavos, aunque el servidor maestro también puede realizar, en un momento dado, alguna operación con archivos básicos.

Otra cosa que se puede observar en la figura es que los archivos son particionados en bloques más pequeños cuando son almacenados en el sistema de archivos distribuido de Hadoop (HDFS).

Las unidades que proporcionan la escalabilidad horizontal a HBase son las llamadas Regiones, que son un subconjunto de datos de la tabla que se almacenan en una gama contigua de filas.

Las funciones de un RegionServer son:

- ✓ Realizar el hosting y gestionar las Regiones
- ✓ Dividir las regiones en caso de ser necesario (auto-sharding)
- ✓ Manipular las solicitudes de escritura y lectura.
- ✓ Establecer la comunicación con el cliente.

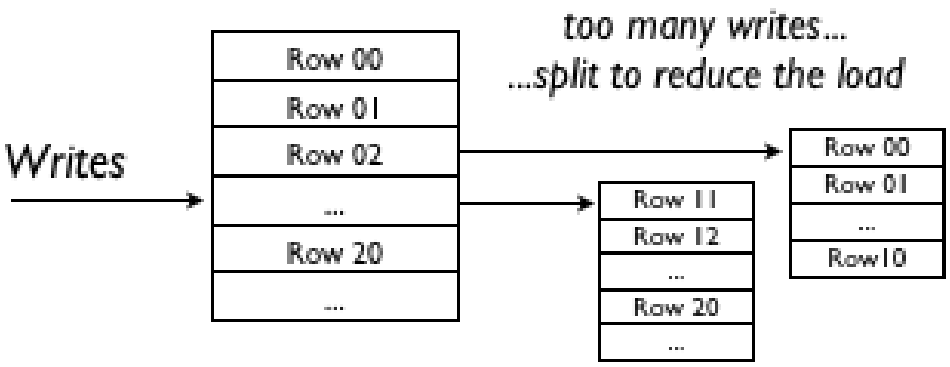
En HBase el servidor maestro es el responsable de la coordinación de las Regiones en el clúster y ejecuta las operaciones administrativas. Un servidor esclavo, también llamado RegionServer, puede servir a una o más regiones. Cada región está asignada a un único RegionServer y el servidor maestro puede decidir mover una región de un RegionServer a otro como resultado de una operación de balanceo de carga. El servidor maestro también se ocupa de gestionar los fallos de los RegionServer, asignando en caso de fallo, la región a otro RegionServer.

El mapeo de las regiones a RegionServer se mantiene en la tabla META. Esta tabla que hasta hace poco estaba localizada en la tabla ROOT (que ya no existe en las últimas versiones de HBASE), ahora está localizada en ZooKeeper.

Al leer META se puede fácilmente identificar qué región contiene lo que se busca. Esto significa que para operaciones de lectura y escritura, el maestro no está involucrado en absoluto, y los clientes pueden ir directamente al RegionServer para obtener los datos.

Cada RegionServer contiene un registro de lectura llamado HLog y varias regiones. Cada región a su vez se compone de MemStore y múltiples StoreFiles (HFile). Los datos se localizan en estos StoreFiles en forma de familias de columnas. El mapeo de las regiones a RegionServer se mantiene, tal y como se ha comentado anteriormente, en la tabla META. Al intentar leer o escribir datos desde HBase, los clientes leen la información requerida de la tabla META localizada en la región. Y se comunican directamente con el servidor de la región correspondiente.

La asignación y distribución de las regiones a RegionServers es automática y en gran parte no requiere intervención por parte del operador. Inicialmente solo hay una región por tabla. Los datos almacenados en la BigTable se encuentran en su RowKey. Ésta es similar a una clave principal de una base de datos relacional. El diseño de un esquema HBase realmente se reduce a optimizar el uso de ese RowKey. Una de las funciones interesantes en HBase es el Auto-Sharding, que simplemente significa que las tablas se distribuyen dinámicamente por el sistema cuando se vuelven demasiado grandes. Cuando las regiones se vuelven demasiado grandes después de añadir más filas, la región se divide en dos creando así dos mitades aproximadamente iguales.



3.2.3.2. Flujo de Datos.

El flujo general comienza cuando un nuevo cliente contacta con el quórum Zookeeper (un grupo independiente de nodos Zookeeper) para encontrar la clave de una fila en particular. Lo hace mediante la recuperación del nombre del servidor (es decir, el nombre de host) que aloja la región ROOT de Zookeeper. Con esa información se puede consultar el servidor que aloja la tabla META. Por último, se consulta la tabla META del servidor para recuperar el servidor que contiene la fila que el cliente está buscando. Una vez que se sabe qué región contiene la fila buscada se almacena en cache la información de dicho servidor y la del correspondiente HRegionServer. Así para las futuras consultas el cliente tiene almacenada en la cache la información y no tendrá la necesidad de consultar la tabla META.

El flujo de escritura comienza cuando el cliente envía una solicitud put (HTable.put) al HRegionServer. El primer paso es escribir el dato en el write-ahead-log (WAL) usando la clase HLog.

El WAL (write-ahead log) es un fichero estándar de Hadoop SequenceFile que almacena instancias de HLogKey. WAL es un "salvavidas" en caso de desastres. Al igual que un registro BIN de MySQL, registra todos los cambios de datos. El WAL es importante en caso de que algo pase con el almacenamiento primario. Así que si el servidor se bloquea o sufre algún daño, se pueden recuperar todos los datos que tenía el servidor hasta el momento del desastre. También significa que si la escritura de la operación en el log falla la operación no se ejecutará sobre el servidor.

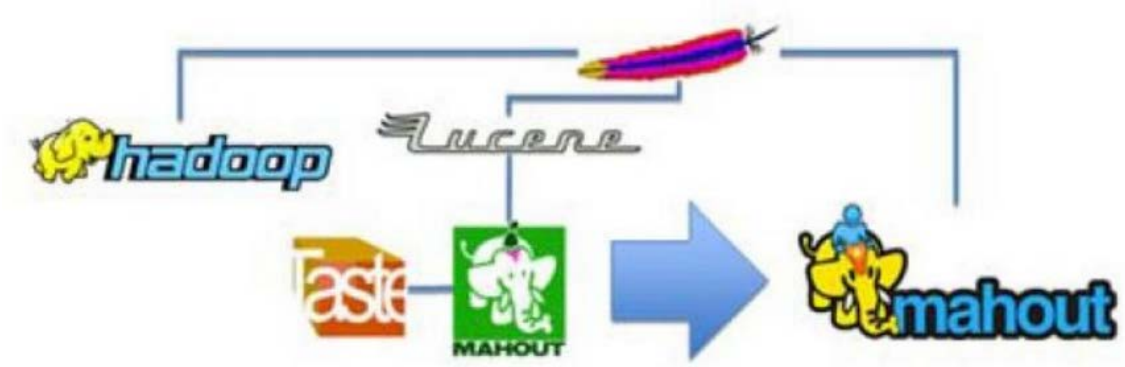
Esta información se utiliza en el caso de que haya un fallo del servidor para saber que registros se deberían haber escrito y no están. Una vez que los datos están escritos en el WAL se proceden a escribirlos en el MemStore. En primer lugar se comprueba si la MemStore está

llena, en el caso de que lo estuviera se utiliza la memoria del disco. Se realiza una petición al HRegionServer con un nuevo hilo que se encarga de escribir los datos en un nuevo HFile situado en el HDFS. También guarda el último número de secuencia para que el sistema sepa los datos existentes hasta el momento. Finalmente cuando el MemStore llega a cierto tamaño o después de un tiempo específico, de forma asíncrona los datos se conservan en el sistema de archivos. Durante todo el proceso los datos se guardan en la memoria volátil.

3.2.4. Apache Mahout.

Apache Mahout es un proyecto de código abierto cuyo objetivo principal es crear un conjunto de bibliotecas de aprendizaje automático diseñadas para ser escalables y robustas. La mayor parte de los algoritmos de Mahout (aunque no todos) están implementados en Mapreduce sobre Hadoop. De ahí viene su nombre, Mahout es una persona que dirige elefantes (el logo de Hadoop es un elefante).

Mahout fue creado inicialmente por Isabel Drost, Grant Ingersoll and Karl Wettin como parte del proyecto Lucene, y llego a ser un top level Project en Abril de 2010.



Una vez que grandes conjuntos de datos se almacenan en el Hadoop Distribute File System (HDFS), Mahout proporciona las herramientas necesarias para encontrar de forma automática patrones significativos en estos grandes conjuntos de datos. El proyecto Apache Mahout pretende que convertir grandes volúmenes de datos en información relevante y útil de una forma rápida y sencilla.

Mahout tiene implementados una gran variedad de algoritmos de aprendizaje automático, algunos en modo local y otros en modo distribuido (para su uso con Hadoop, o

plataformas de reciente aparición que son más adecuadas para el aprendizaje automático como Apache Sparks), que tratan de dar solución a cuatro casos de uso dentro de la ciencia de datos. Estos cuatro casos de uso son:

3.2.4.1. Collaborative filtering.

Trata sobre análisis del comportamiento del usuario y hacer recomendaciones de productos. Es una técnica que popularizó Amazon entre otros y que usa información de los usuarios como puede ser ratings, clicks, compras, etc… para proporcionar recomendaciones basadas en estos datos a otros usuarios.

Existen cuatro formas típicas de generar recomendaciones:

- User-based. Recomienda productos encontrando similitudes entre usuarios. Es decir, encuentra usuarios similares al que está usando la aplicación y le recomienda los productos o artículos que hayan comprado o hayan dado una buena puntuación. Esta forma de generar recomendaciones es difícil de escalar por la gran diversidad de usuarios que existe y porque los gustos de los usuarios van cambiando, no son algo fijo.

- Item-Based. Calcula el nivel de similitud entre artículos. Dado que los artículos o productos no suelen cambiar mucho, este proceso se puede hacer fuera de línea, es decir en batch, teniendo los datos pre calculados cuando un usuario esté navegando por la aplicación. Por ejemplo, si un usuario está mirando en una web un libro, este algoritmo podría, previamente haber calculado varios productos similares a este libro (misma temática, misma editorial, o cualquier otra característica), y por lo tanto, la web podría sugerirle al usuario estos productos mientras está mirando el libro.

- Slope-One. Algoritmo predictivo basado en predecir las puntuaciones de un artículo en función de las puntuaciones que se le han dado a otros artículos.El algoritmo es muy sencillo, y fácil de programar y es rápido y bastante fiable comparado con otros mucho máas complejos. El algoritmo consiste básicamente en calcular la diferencia media entre las puntuaciones que los usuarios hayan dado a dos artículos, habiendo sido puntuados ambos por los usuarios. Con este cálculo se podría predecir la nota que le daría un usuario al artículo B, sumando la media a la puntuación que le haya dado al artículo A.

A continuación vamos a revisar un ejemplo ilustrativo.

USER	HARRY POTTER	BATMAN	SPIDERMAN
U1	5	3	4
U2	?	2	4
U3	4	2	?

Como funcionaría Slope-One para calcular la puntuación que le daría el usuario número tres a Spiderman.

Primero obtendría la media de las diferencias entre Batman y Spiderman:

$$[(4-3)+(4-2)]/2 = 1,5$$

La puntuación de Spiderman menos la de Batman para el usuario uno, más la puntuación de Spiderman menos la de Batman para el usuario dos, dividido entre dos usuarios. El resultado es 1,5.

La puntuación que daría el usuario tres a Spiderman usando este resultado sería igual a la puntuación de Batman más la desviación media

$$2 + 1,5 = 3,5$$

De la misma forma se podría hacer para Harry Potter y Spiderman.

$$[(4-5)]/1 = -1$$

La puntuación que daría el usuario tres a Spiderman usando este resultado sería igual a la puntuación de Harry Potter más la desviación media

$$4 - 1 = 3$$

Para ajustar más el resultado se pueden considerar ambas predicciones, ponderándolas en función del número de usuarios con el que se hayan obtenido.

$$(3,5 * 2 + 3 * 1)/(2 + 1) = 3,33$$

- Model-Based. Este algoritmo proporciona recomendaciones basadas en el desarrollo de un modelo de usuarios y sus ratings.

3.2.4.2. Clustering.

Teniendo un gran conjunto de datos, ya sean de tipo texto o numéricos, a menudo es útil agruparlos automáticamente en función de las similitudes que tengan sus elementos. Por ejemplo, dadas todas las noticias existentes en un día de todos los periódicos que hay en España podría ser útil agrupar automáticamente todas las noticias que tratan sobre un mismo tema juntas, para de esta forma poder concentrarse en el estudio de un tema sin tener que leer un montón de noticias que no están relacionadas con ese tema.

Otro ejemplo podría ser, teniendo las salidas de los sensores de una máquina, podría ser útil agrupar las lecturas de funcionamiento normal, y aquellas que determinen un comportamiento anormal de la máquina.

Tal y como hace Collaborative Filtering, Clustering calcula las similitudes entre los diferentes ítems de un conjunto de datos, pero su único trabajo es agruparlos en función de esas similitudes.

En muchas de las implementaciones de clustering, los ítems de una colección son representados por vectores en un espacio de n dimensiones de forma que dados dos vectores, se puede calcular la distancia que hay entre dos ítems usando medidas como "*Manhattan distance*", "*Euclidean distance*" o "*cosine similarity*", entonces las agrupaciones pueden ser calculadas en función a esta distancia.

Existen muchas formas de calcular los cluster o agrupaciones, siendo alguna de las más conocidas "*k-means*" o "*hierarchical clustering*".

3.2.4.3. Classification:

También llamado "*Categorization*", tiene como meta el etiquetado de documentos nuevos basándose en la clasificación de otros ya etiquetados, para de esta forma agruparlos juntos. Muchos enfoques de clasificación en el aprendizaje automático calculan una variedad de estadísticas que asocian características de un documento con una determinada etiqueta y así pueden crear un modelo que después puede ser usado para clasificar documentos que aún no han sido vistos. Por ejemplo, un enfoque simple de clasificación podría intentar rastrear las palabras asociadas a una determinada etiqueta, así como el número de veces que aparecen estas palabras para poder asignar una etiqueta a un documento. De esta forma, cuando un nuevo documento es clasificado, las palabras del documento se buscan en el modelo de clasificación, calculando las probabilidades de encajar más en una etiqueta o en otra y

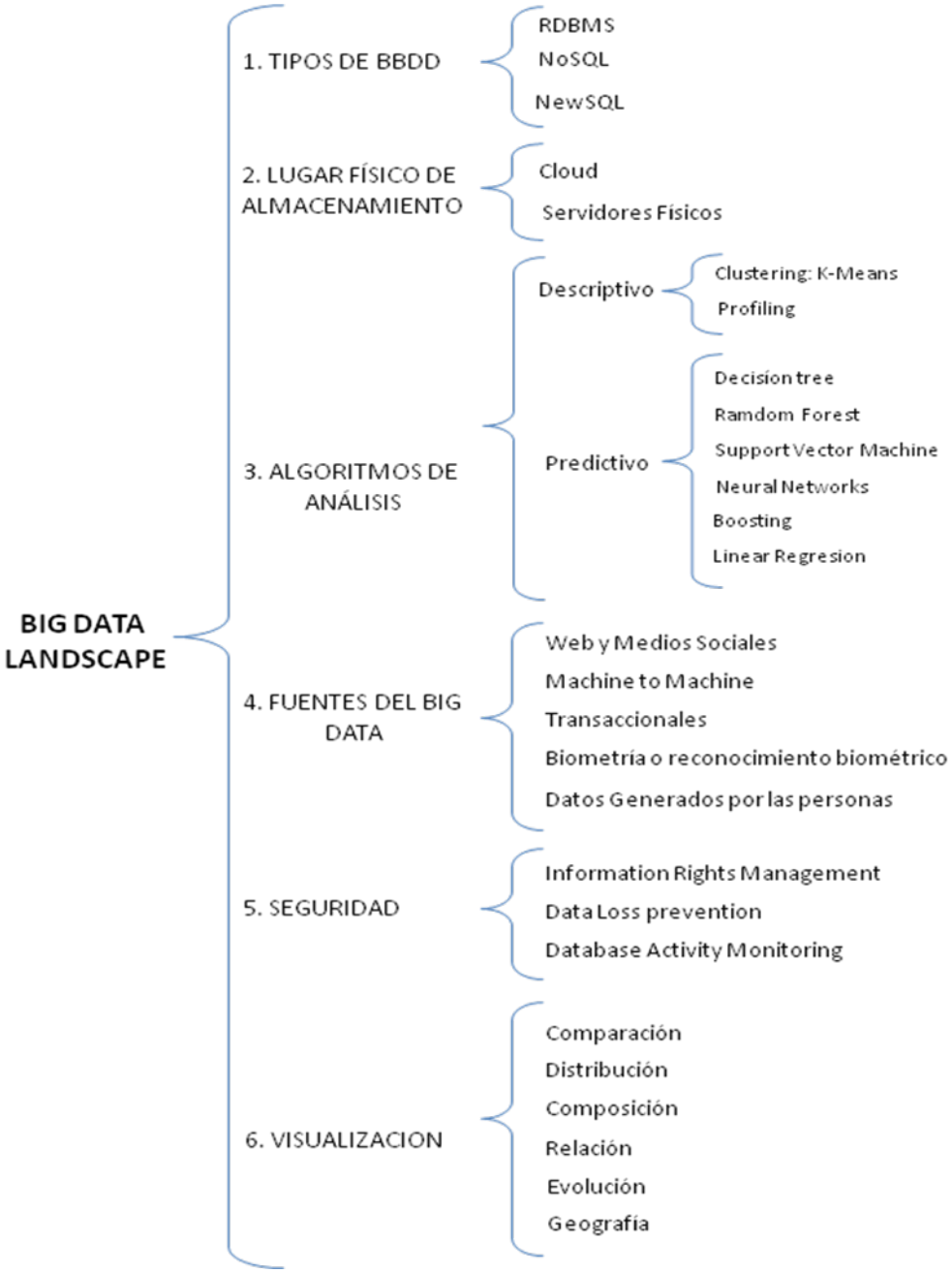
se obtiene como salida el mejor resultado, normalmente con un grado de confianza que indica la fiabilidad de que el resultado pueda ser correcto.
Las características usadas para la clasificación, pueden incluir palabras, pesos de estas palabras (basados en la frecuencia, por ejemplo), partes del documento y muchas otras, en realidad, cualquier característica que pueda ayudar a clasificar el documento puede ser incluida dentro del algoritmo de clasificación.

3.2.4.4. Frecuent Itemset Mining:

Este algoritmo se basa en el análisis de cómo se agrupan los artículos, por ejemplo, el contenido de un carro de la compra, de forma que pueda identificar que artículos, de forma individual, normalmente aparecen juntos.

4. LANDSCAPE BIG DATA ALTERNATIVO.

Después de la descripción que se ha hecho de uno de los landscape existentes, para ayudar a comprender mejor las diferentes posibilidades que proporciona este nuevo mundo del big data, a través de una pequeña descripción a alto nivel el funcionamiento de las principales tecnologías que usa, el siguiente paso es poder realizar un landscape propio, atendiendo a una clasificación original a la que se ha ido dando forma mientras se analizaba toda la información proporcionada en los puntos anteriores y que se pasa a detallar a continuación.



4.1.TIPOS DE BASES DE DATOS.

Una de las decisiones que se tienen que tomar al desarrollar un sistema de análisis de big data es la base de datos en la que se almacenará la información. En este sentido, se pueden dividir principalmente en tres tipos:

4.1.1. RDBMS (Relational DataBase Management System).

Las bases de datos relacionales aparecieron en 1970 cuando Edgar Frank Codd de IBM publicó su trabajo "A Relational Model of Data for Large Shared Data Banks".

Una Base de Datos relacional es una base de datos digital que se organiza basándose en el modelo relacional. Este modelo organiza los datos en tablas de filas y columnas, con una clave primaria única para cada fila.

Generalmente, cada entidad u objeto que quiere representarse en una base de datos tiene su propia tabla, con su propia clave primaria, las filas de la tabla (también llamadas registros) representan instancias de esa entidad u objeto, y las columnas representan atributos de esa entidad.

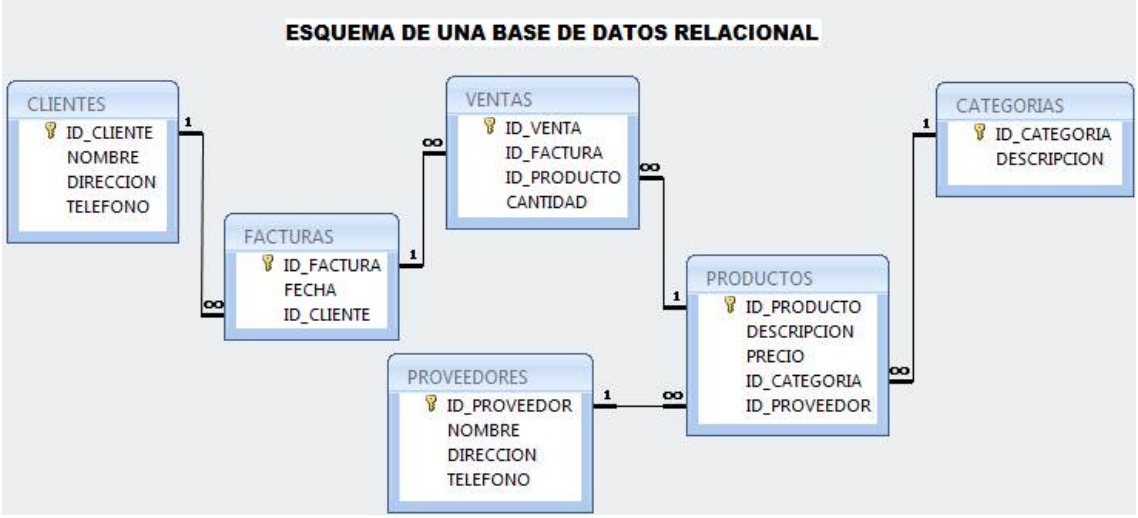
A continuación se van a comentar varios términos que caracterizan a todas las bases de datos relacionales:

- Tabla. Básicamente es una colección de elementos organizados en filas y columnas como se ha comentado en el párrafo anterior. También se la puede describir como una representación adecuada de las relaciones. Es la forma más simple de almacenamiento de datos.
- Registro. Una única entrada de una tabla se la denomina registro o fila. Un registro de una tabla representa un conjunto de datos relacionado.
- Columna. En una tabla relacional, una columna en un conjunto de valor de un tipo en particular. También se le suele llamar atributo.
- Clave primaria. Es aquella de entre todas las claves candidatas que es elegida identifica unívocamente un registro dentro de la tabla relacional.
- Clave foránea. Es aquella que hace referencia a la clave primaria de otra tabla. Estas claves son las que definen las relaciones entre las diferentes tablas de la base de datos. Su propósito es asegurar la integridad referencial. Esta clave debe ser única en la tabla a la que referencian, pero no hace falta que lo sean en la tabla en la que están. Por ejemplo, el código de cliente puede ser clave foránea en la tabla de pedidos, lo que indica que un pedido puede pertenecer a un cliente, pero un mismo cliente podría tener varios pedidos.
- Índice. Es aquella estructura de datos definida sobre una o varias columnas de una tabla y que permiten un acceso más rápido a los datos.
- Restricción. Son limitaciones o restricciones que obligan a la base de datos a cumplir ciertas condiciones. Algunas pueden estar definidas a nivel de columna, por ejemplo, que no pueda tener valores negativos y otras son inherentes al hecho de que la base de datos sea realcional, por ejemplo, una tabla no puede tener dos registros con la misma clave primaria.

- Procedimientos almacenados. Son programas o funciones que están codificadas para acceder a la base de datos y que se almacenan en la misma base de datos. Generalmente son operaciones que se tengan que hacer habitualmente en la base de datos.

Otro concepto importante a la hora de definir una base de datos relacional es la estructura, es decir, como se organiza la base de datos. Las bases de datos relacionales se organizan en dos secciones:

- Esquema. Un esquema es básicamente un contenedor en el que se agrupan tablas y otros objetos de la base de datos relacional. Esquemas típicos en una base de datos son dbo, temp o system. Un esquema no puede contener a otro, simplemente divide la base de datos en partes, pero no hay "subpartes" dentro de éstas.
- Datos o instancia. Es el estado de la base de datos en un momento concreto, es decir, el contenido de la base de datos.



Para que una base de datos relacional funcione eficientemente y de forma precisa debe trabajar con lo que en inglés se denomina "ACID transactions":

- Atomicity (Atomicidad). Es la propiedad que asegura que una operación de una base de datos se ha realizado o no, es decir, que si hay algún error en algún punto de la transacción, ésta es cancelada por completo, asegurando que no pueda quedarse a medias.

- Consistency (Consistencia o Integridad). Es la propiedad por la cual se asegura que solamente datos válidos pueden ser escritos en la Base de Datos. Es decir, cuando se ejecuta una transacción y esta finaliza, si es de forma correcta, los datos deben ser válidos y no romper ninguna de las reglas definidas en la base de datos, si es de forma errónea la base de datos revierte todos los cambios para volver al estado válido inicial.
- Isolation (Aislamiento). Esta propiedad es la que asegura que dos transacciones no pueden afectarse una a la otra, es decir, las transacciones están aisladas unas de otras, aun ejecutándose en paralelo, no deben afectarse unas a otras.

- Durability (Durabilidad o Persistencia). Esta propiedad asrgura que cualquier transacción finalizada (commit realizado en la base de datos) no se perderá. Esto se garantiza mediante los backup y los log de transacciones que facilitan la restauración de cualquier transacción confirmada a pesar del error de cualquier software o hardware de la base de datos.

Para manipular la información de una base de datos se usa un lenguaje relacional. Actualmente existen dos lenguajes formales que son:

- Algebra relacional. Que permite describir la forma de realizar una consulta.
- Cálculo relacional. Que únicamente indica lo que se desea devolver.

El lenguaje más común en las bases de datos relacionales a la hora de crear consultas es el SQL (Structured Query Leguage).

Entre los sistemas de gestión de bases de datos relacionales más usados hoy en día están por ejemplo:

- Sybase.
- DB2.
- MySQL
- Microsoft SQL Server.
- Oracle.
- Informix

4.1.2. NoSQL (No solo SQL).

Son un grupo de sistemas de gestión de bases de datos que difieren en varios aspectos de los tradicionales sistemas de gestión de bases de datos relacionales.

La primera vez que apareció este término fue en 1998 cuando Carlo Strozzi lo uso para referirse a su base de datos open source, que era una base de datos que no ofrecía un interface SQL, aunque sí que seguía el modelo relacional. Más adelante, Rick Evans lo reutilizó para referirse al creciente número de bases de datos no relacionales y distribuidas que no garantizaban el ACID.

Así como las bases de datos relacionales se caracterizan por las propiedades transaccionales ACID (Atomicity, Consistency, Isolation, Durability), las bases de datos NoSQL a menudo son relacionadas con el acrónimo BASE:

- Basically Available. Estos sistemas usan la replicación de datos y el uso de lo que se denomina "sharding" que básicamente consiste en dividir los datos en varios servidores de almacenamiento diferentes. Estas dos características hacen que los sistemas NoSQL estén siempre disponibles, incluso si subconjuntos de datos no están disponibles por cortos periodos de tiempo.
- Soft State. Mientras que los sistemas ACID asumen que la consistencia de datos es un requisito imprescindible, los sistemas NoSQL permiten que los datos sean inconsistentes y relegan el diseño encargado de gestionar tales inconsistencias a los desarrolladores de aplicaciones.
- Eventually Consistent. En contraste con los sistemas ACID que aseguran la consistencia en cada commit de una transacción, los sistemas NoSQL garantiza la coherencia de los datos en algún momento futuro indeterminado.

Los sistemas NoSQL surgieron para cubrir las necesidades que comenzaban a tener empresas como Amazon, Google, Linkedin o Twiter que tenían que lidiar cada vez con mayor número de operaciones y volumen de datos, así como análisis de grandes volúmenes de datos en tiempo real, todo ello buscando una ventaja competitiva mediante el aprovechamiento de esta información alojada tanto en datos no estructurados, como semiestructurados.

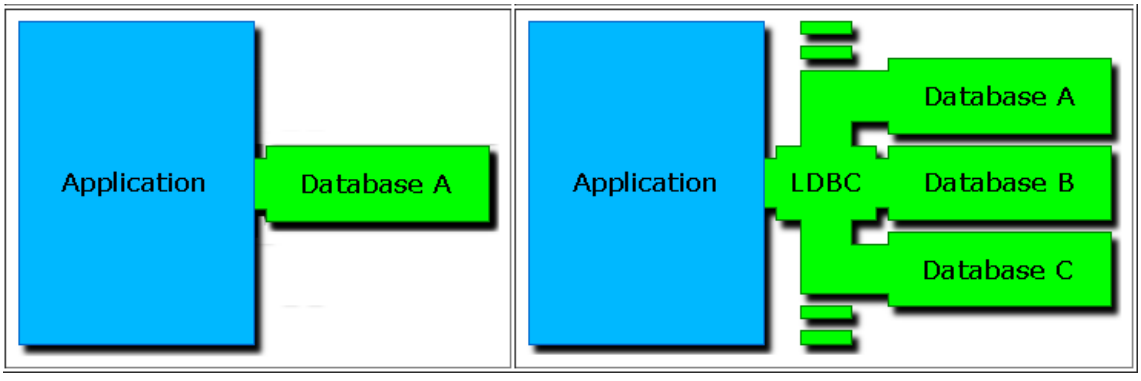
Las bases de datos relacionales no daban el rendimiento necesario, por lo que este tipo de empresas invirtieron mucho en la investigación y desarrollo de este tipo de sistemas de bases de datos.

4.1.3. NewSQL (Nuevo SQL).

El término apareció en 2011 en un documento de investigación que publicó el analista del grupo 451, Matthew Aslett, en el que hablaba de la creciente necesidad de empresas de gestionar una gran cantidad de datos, estructurados y no estructurados, cada vez en menor tiempo, pero que no podían usar NoSQL porque para ellas era imprescindible tener consistencia transaccional (por ejemplo, sistemas financieros). La única solución hasta el momento para estas empresas era comprar máquinas más potentes o desarrollar soluciones a medida que distribuyeran las queries en diferentes nodos, sin embargo, cualquiera de estas era demasiado cara para la mayoría de ellas y por lo tanto esta no era una opción.

Podemos considerar NewSQL a la clase de sistemas de gestión de bases de datos relacionales que buscan proporcionar el mismo rendimiento escalable de los sistemas NoSQL, pero manteniendo las propiedades ACID de los sistemas de bases de datos tradicionales.

Otro de los problemas que soluciona NewSQL es la estandarización del lenguaje. SQL no es realmente estándar, ya que cada producto tiene sus pequeñas diferencias en las reglas de sintaxis. Esto lo soluciona NewSQL con LDBC (Liberty DataBase Connectivity), un driver que proporciona acceso a las bases de datos traduciendo de SQL a NewSQL y a la inversa. Gracias a este conector basado principalmente en ANSI-SQL, una aplicación puede trabajar con la mayor parte de bases de datos sin necesidad de cambiar su código fuente.



En la figura se puede ver cuál sería la diferencia entre una aplicación que solo puede acceder a una base de datos, y esa misma aplicación sin modificar el código fuente,

usando LDBC como intermediario, puede conectarse a varias bases de datos diferentes.

Los sistemas NewSQL son clasificados generalmente en tres categorías que se detallan a continuación:

- Nuevas arquitecturas. Son plataformas de bases de datos completamente nuevas. Están diseñadas para trabajar en un cluster distribuido en el que cada nodo posee un subconjunto de los datos.

  Ejemplos de sistemas de gestión de bases de datos pertenecientes a esta clasificación son Google Spanner, Clustrix o NuoBD.
- Motores SQL. Esta segunda categoría son motores de almacenamiento altamente optimizados para SQL. Estos sistemas proporcionan el mismo interface de programación que SQL pero con una mejor escalabilidad. Ejemplos de sistemas de gestión de bases de datos pertenecientes a esta clasificación son InfoBright o TokuBD
- Sharding Transparente. Estos sistemas proporcionan una capa intermedia que divide automáticamente las bases de datos a lo largo de múltiples nodos. Ejemplos de sistemas de gestión de bases de datos pertenecientes a esta clasificación son dbShards, Scalearc o ScaleBase.

4.2.LUGAR FÍSICO DE ALMACENAMIENTO.

Hoy en día los datos son un gran negocio. Tanto las grandes empresas como las pequeñas startups recopilan, almacenan y gestionan grandes cantidades de datos de clientes, proveedores, empleados y estadísticas de marketing (big data).

Todos estos datos necesitan almacenarse en algún sitio y es importante para las empresas poder tomar una buena decisión en relación a este tema. Para ello necesitan conocer los pros y los contras tanto de tener o comprar un servidor local, como de decantarse por una solución basada en almacenamiento en la nube.

4.2.1. Servidor Local.

A continuación se describirán los pros y los contras de tener los datos en un servidor local tradicional.

Ventajas.

- La seguridad local puede ser controlada por el equipo de tecnología de la empresa propietaria del servidor.
- Posibilidad de ejecutar cualquier aplicación, sin ninguna restricción, ya que el control sobre el servidor es absoluto.
- El acceso a los ficheros, backups e impresoras es más rápido.
- Acceso al hardware real en caso de fallo.
- No se necesita conexión a internet, ni depende de la velocidad de esta conexión para acceder a los datos.

Inconvenientes.

- El capital inicial es elevado, ya que comprar un servidor es caro.
- Los costes de mantenimiento pueden ser altos si se necesitan tener varios servidores.
- Seguramente sea necesario contratar a un equipo de IT para mantener el hardware y el software de los servidores.
- Si el crecimiento de la empresa es rápido, es posible que se quede pequeño el servidor y se tenga que comprar otro.

4.2.2. Cloud.

En cuanto a la solución basada en el alquiler de servicios de almacenamiento en la nube, también tiene sus 'pro' y sus 'contra', que podemos enumerar a continuación.

Ventajas.

- No se necesita una gran inversión inicial, y los costes mensuales son estables.
- La expansión de tus servidores en función de tus necesidades (por ejemplo, rápido crecimiento de la empresa) es ilimitada, solo se necesitaría contratar un servicio mayor.
- No se tienen gastos de mantenimiento.
- Posibilidad de acceder a la información desde cualquier sitio (siempre que exista conexión a internet) sin ningún coste adicional.

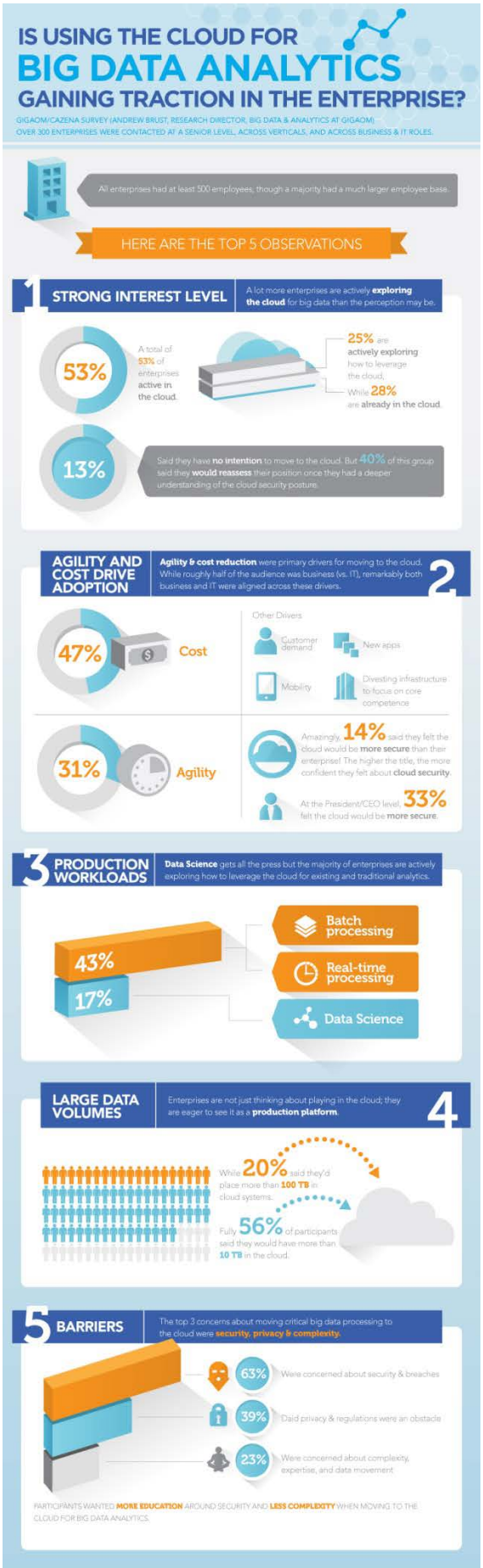
Inconvenientes.

- No se tiene la posibilidad de ejecutar cualquier software que se desee, dependerá del proveedor de servicios en la nube.
- Los datos se almacenarán físicamente en el centro de datos del proveedor
- Cambiar de proveedor de servicios en la nube, una vez que se tiene uno, es muy complicado.
- El acceso a los datos puede ser complicado o lento, ya que depende la conexión a internet.

Ninguna de las dos soluciones es mejor que la otra, la mejor solución dependerá mucho de las necesidades y posibilidades propias de cada empresa.

4.2.3. Estadísticas.

A continuación, se muestra una imagen en la que se ven varios datos estadísticos con los que se trata de representar tanto el uso que se le da a la nube y a big data en el mundo empresarial.


IS USING THE CLOUD FOR
BIG DATA ANALYTICS
GAINING TRACTION IN THE ENTERPRISE?
GIGAOM/CAZENA SURVEY (ANDREW BRUST, RESEARCH DIRECTOR, BIG DATA & ANALYTICS AT GIGAOM)
OVER 300 ENTERPRISES WERE CONTACTED AT A SENIOR LEVEL, ACROSS VERTICALS, AND ACROSS BUSINESS & IT ROLES.
All enterprises had at least 500 employees, though a majority had a much larger employee base.
HERE ARE THE TOP 5 OBSERVATIONS
1 STRONG INTEREST LEVEL
A lot more enterprises are actively exploring the cloud for big data than the perception may be.
53%
A total of 53% of enterprises active in the cloud.
25% are actively exploring how to leverage the cloud,
While 28% are already in the cloud.
13%
Said they have no intention to move to the cloud. But 40% of this group said they would reassess their position once they had a deeper understanding of the cloud security posture.
AGILITY AND COST DRIVE ADOPTION
2
Agility & cost reduction were primary drivers for moving to the cloud. While roughly half of the audience was business (vs. IT), remarkably both business and IT were aligned across these drivers.
47%
Cost
Other Drivers
Customer demand
New apps
Mobility
Divesting infrastructure to focus on core competence
31%
Agility
Amazingly, 14% said they felt the cloud would be more secure than their enterprise! The higher the title, the more confident they felt about cloud security.
At the President/CEO level, 33% felt the cloud would be more secure.
3 PRODUCTION WORKLOADS
Data Science gets all the press but the majority of enterprises are actively exploring how to leverage the cloud for existing and traditional analytics.
43%
17%
Batch processing
Real-time processing
Data Science
LARGE DATA VOLUMES
4
Enterprises are not just thinking about playing in the cloud; they are eager to see it as a production platform.
While 20% said they'd place more than 100 TB in cloud systems.
Fully 56% of participants said they would have more than 10 TB in the cloud.
5 BARRIERS
The top 3 concerns about moving critical big data processing to the cloud were security, privacy & complexity.
63%
Were concerned about security & breaches
39%
Said privacy & regulations were an obstacle
23%
Were concerned about complexity, expertise, and data movement
PARTICIPANTS WANTED MORE EDUCATION AROUND SECURITY AND LESS COMPLEXITY WHEN MOVING TO THE CLOUD FOR BIG DATA ANALYTICS.

4.3. ALGORITMOS DE ANÁLISIS.

En el mundo del Big Data, además de gestionar el problema de almacenamiento de grandes cantidades de información, ha desarrollado herramientas de análisis de esta información que le dan un valor añadido a esta.

A continuación se enumerarán diferentes clases de algoritmos existentes para la explotación de esta información, clasificados principalmente en dos tipos, descriptivos y predictivos.

4.3.1. Algoritmos Descriptivos.

Los algoritmos descriptivos son técnicas matemáticas que permiten descubrir patrones o correlaciones entre las grandes cantidades de datos almacenados en las bases de datos de los sistemas de big data.

Los dos principales algoritmos descriptivos son:

4.3.1.1. Data Profiling.

Es un proceso que trata de analizar los datos almacenados para recopilar información y estadísticas sobre estos datos y cuya finalidad puede ser:

- Intentar averiguar si estos datos pueden ser fácilmente utilizables para algún otro propósito.
- Mejorar la capacidad de búsqueda de información en estos datos mediante el etiquetado con palabras clave, descripciones o asignación a una categoría.
- Evaluar el riesgo que implica la integración de los datos en nuevas aplicaciones.
- Evaluar si los metadatos describen con precisión los valores actuales en la base de datos.
- Tener una visión corporativa de los datos para diferentes usos como la gestión de los datos, o el gobierno de los datos para mejorar la calidad de los mismos.

El análisis de datos mediante Data Profiling se usa por ejemplo en las empresas para analizar si las fuentes de datos candidatas para integrar en la base de datos son válidas, y para aclarar la estructura, el contenido, las relaciones y reglas de derivación de los datos.

No sólo ayuda a entender las anomalías y para evaluar la calidad de los datos, sino también para descubrir, registrar y evaluar metadatos de los datos

analizados. En este sentido, el propósito del data profiling es a la vez para validar los metadatos cuando están disponibles y descubrir metadatos nuevos cuando no lo están. El resultado del análisis es usado por la empresa de forma estratégica, para determinar la idoneidad de los sistemas de origen candidatos para alimentar su datawarehaouse, y tácticamente, para identificar posteriormente los problemas de diseño de la solución que se adoptará en el proyecto de integración de los datos, y también para controlar las expectativas de la empresa.

Este tipo de análisis usa diferentes tipos de estadísticas descriptivas tales como:

- Mínimo.
- Máximo.
- Media.
- Moda.
- Percentil.
- Desviación estándar.
- Frecuencia.
- Variación.
- Otros métodos como Suma o recuento

Los metadatos obtenidos mediante este tipo de análisis son por ejemplo:

- Tipo de datos.
- Longitud.
- Singularidad.
- Valores discretos.
- Aparición de valores nulos.
- Patrones de cadenas de caracteres típicos.

Estos metadatos pueden usarse posteriormente para descubrir problemas como valores ilegales, fallos ortográficos, duplicados, etc…

Se realizan diferentes análisis para evaluar diferentes niveles estructurales, por ejemplo:

- Se pueden realizar análisis de cada columna de forma independiente para entender la frecuencia de distribución de los diferentes valores, tipos y uso de cada columna.

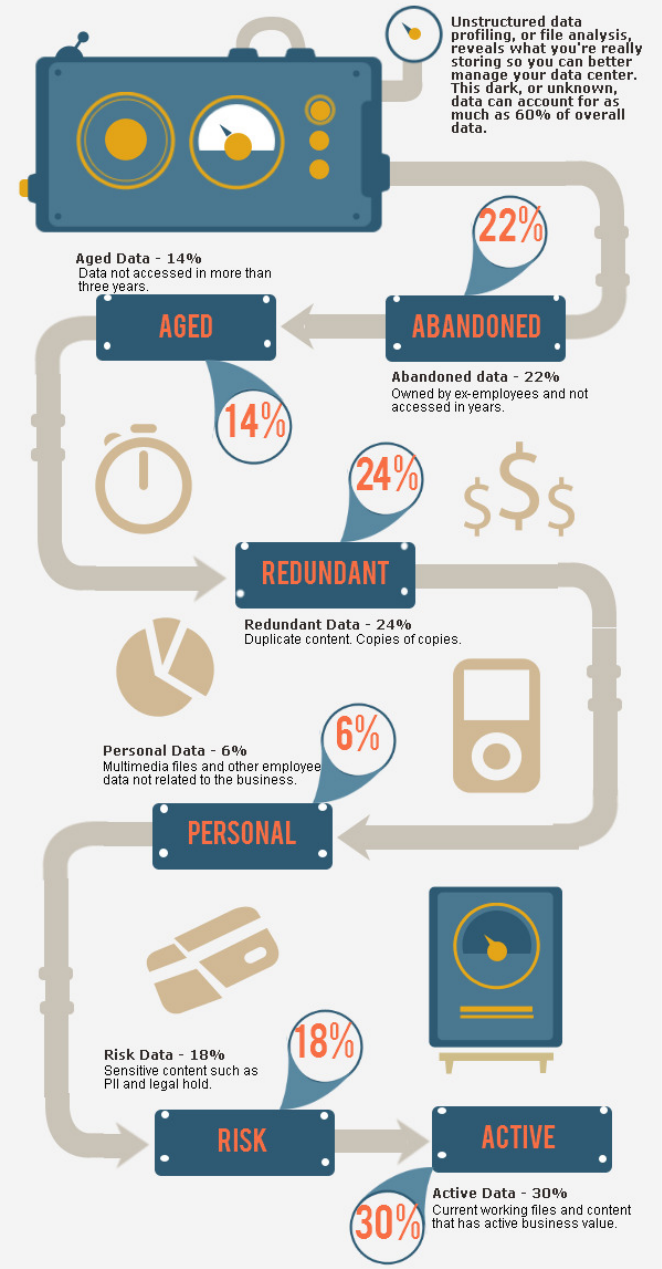
- Dependencias intrínsecas entre valores de diferentes columnas pueden ser descubiertas en un análisis de varias columnas de forma transversal.
- Por último, también pueden encontrarse relaciones de tipo clave foránea en el análisis conjunto de varias tablas.

Normalmente las herramientas de análisis a través de data profiling se usan para facilitar un proceso posterior, ya sea de análisis o de explotación de los datos.

La complejidad de cálculo de estos análisis aumenta en función del tipo de análisis, siendo los más sencillos los análisis de una única columna y los más complejos los análisis de varias tablas de forma conjunta.

Los beneficios visibles de este tipo de análisis son por lo tanto mejorar la calidad de los datos, acortar el ciclo de ejecución de los proyectos de integración de información en el datawarehouse de la empresa y mejorar la comprensión de los datos a los usuarios de los mismos.

A continuación se puede ver el resultado de un análisis de datos almacenados realizado en una empresa ficticia mediante data profiling de forma que podemos ver como clasifica y filtra la información, para facilitar su uso posterior.



4.3.1.2. K-Means.

K-Means es uno de los algoritmos de aprendizaje más sencillos que resuelve el problema de la agrupación de datos.

El procedimiento sigue una manera simple y fácil de clasificar un conjunto de n observaciones a través de un cierto número de grupos de datos dado (asumir racimos k) fijado a priori.

La idea principal es definir los centros de k, también llamados centroides, uno para cada grupo. Estos centros deben ser colocados eficientemente, ya que el resultado del algoritmo puede variar en función de esta primera elección.

Existen dos formas de elegir esta primera asignación de la posición de los centroides:

- La primera se denomina método de Forgy que simplemente elige aleatoriamente k (k=número de grupos) observaciones para usarlas como primeros centroides.
- La segunda es el método de partición aleatoria, que trata de asignar un grupo inicial para cada observación.

La mejor opción es colocar los centroides tan lejos como sea posible unos de los otros.

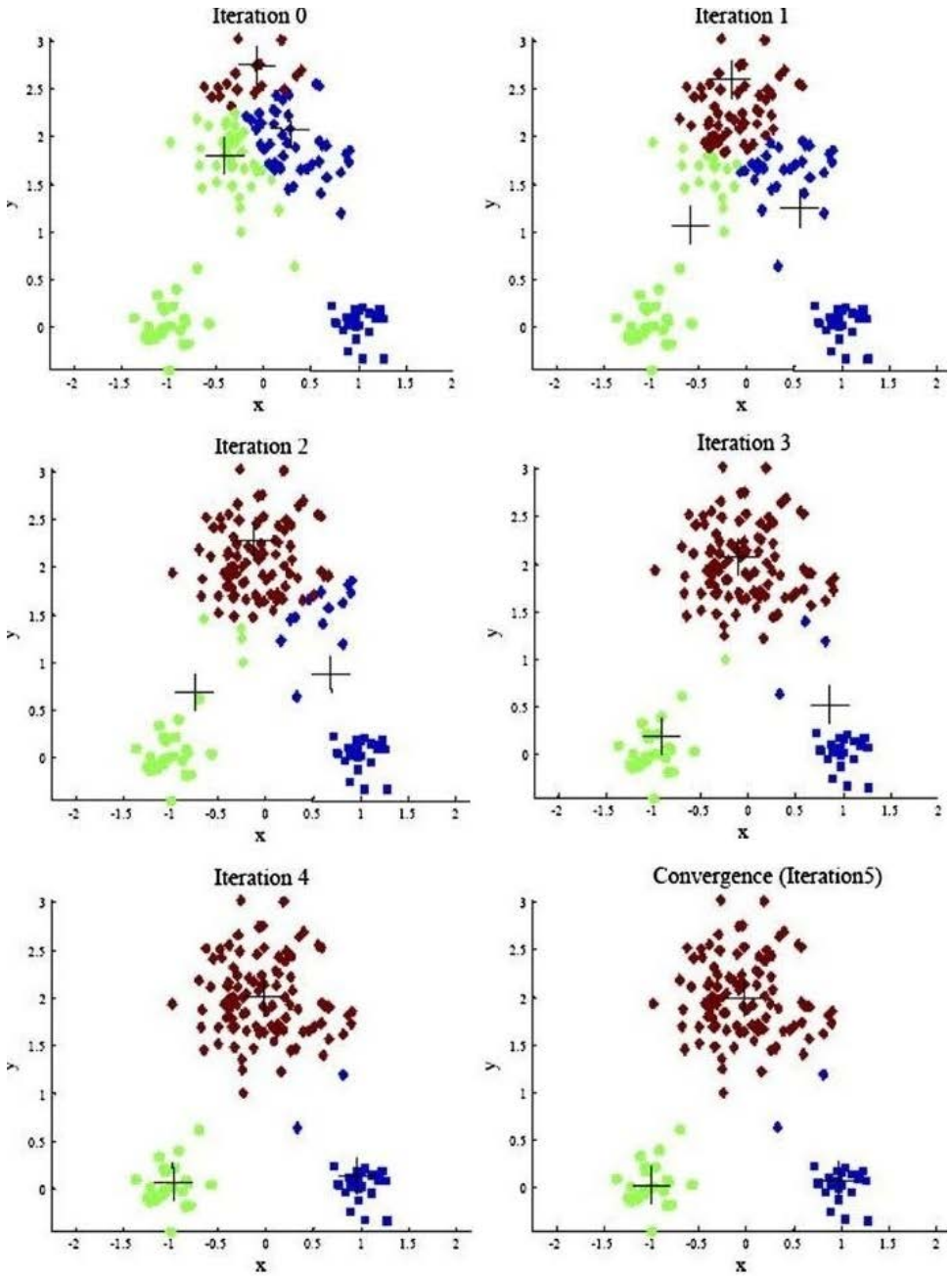
El siguiente paso es tomar cada punto de la pertenencia a un determinado conjunto de datos y asociarlo al centro más cercano. Cuando todas las observaciones están asignadas a un grupo, el primer paso se ha completado. En este punto tenemos que volver a calcular k nuevos centroides como baricentro de los grupos resultantes de la etapa anterior.

Después, una vez que tenemos estos nuevos centroides k, se deben volver a clasificar las observaciones, cada una perteneciendo al grupo más cercano. Estos pasos son iterativos hasta que los centroides no cambian de posición de un paso a otro.

En resumen, los pasos a seguir para la ejecución de este algoritmo son:

- Realizar la primera selección de centroides.
- Asignar cada observación al centroide más cercano.
- Cuando todas las observaciones están asignadas, calcular de nuevo el centroide de cada grupo.
- Repetir los dos pasos anteriores hasta que los centroides dejen de recolocarse.

Aunque el algoritmo siempre termina, no siempre tiene porque alcanzar la solución óptima. El algoritmo es bastante sensible a la primera selección de centroides.



Es esta imagen se puede ver un ejemplo de clasificación a través del algoritmo de K-Means.

Las ventajas de este algoritmo son:

- Rápidez
- Es robusto
- Fácil de entender.
- Realtivamente eficiente. Para N obseravaciones, k grupos, d dimensiones y t iteraciones, normalmente se cumple que N >> k, d, t

- Proporciona el mejor resultado en aquellos casos en los que los datos son muy diferentes y están bien separados unos de otros.

Las desventajas de este algoritmo son:

- Se debe conocer a priori el número k de grupos en los que se va a clasificar la información.
- El algoritmo de aprendizaje puede variar con transformaciones no lineales, por ejemplo, si los datos se representan de diferente forma el resultado cambia, es decir, el resultado no es el mismo cuando representamos los datos en coordenadas cartesianas o en coordenadas polares.
- El algoritmo falla para conjuntos de datos no lineales.
- El algortimo no gestiona bien loa valores atípicos.
- El azar en la primera elección de centroides puede que el resultado final no sea el óptimo.

4.3.2. Algoritmos Predictivos.

El análisis predictivo se compone de varias técnicas:

- Técnicas de estadísticas de modelado.
- Técnicas de aprendizaje automático.
- Técnicas de minería de datos.

Que se usan para analizar datos del pasado y del presente, para predecir eventos del futuro.

Estas técnicas se usan en muchos campos como seguros, banca, medicina, etc… pero para lo que más se están usando es para marketing y ventas.

Las campañas comerciales o publicitarias aumentan su efectividad notablemente usando el análisis predictivo, cada vez se es más fácil acertar y conocer a los clientes y usuarios, incluso intentar saber cuál será el siguiente movimiento de la competencia.

En los últimos años, además de en estos campos, se está empezando a usar el análisis predictivo también para mejorar internamente la empresa. El análisis predictivo puede tener utilidad optimizando procesos internos, minimizando riesgos, incluso asegurando la calidad.

Existe una gran variedad de algoritmos de análisis predictivo, a continuación se describirán algunos de forma básica:

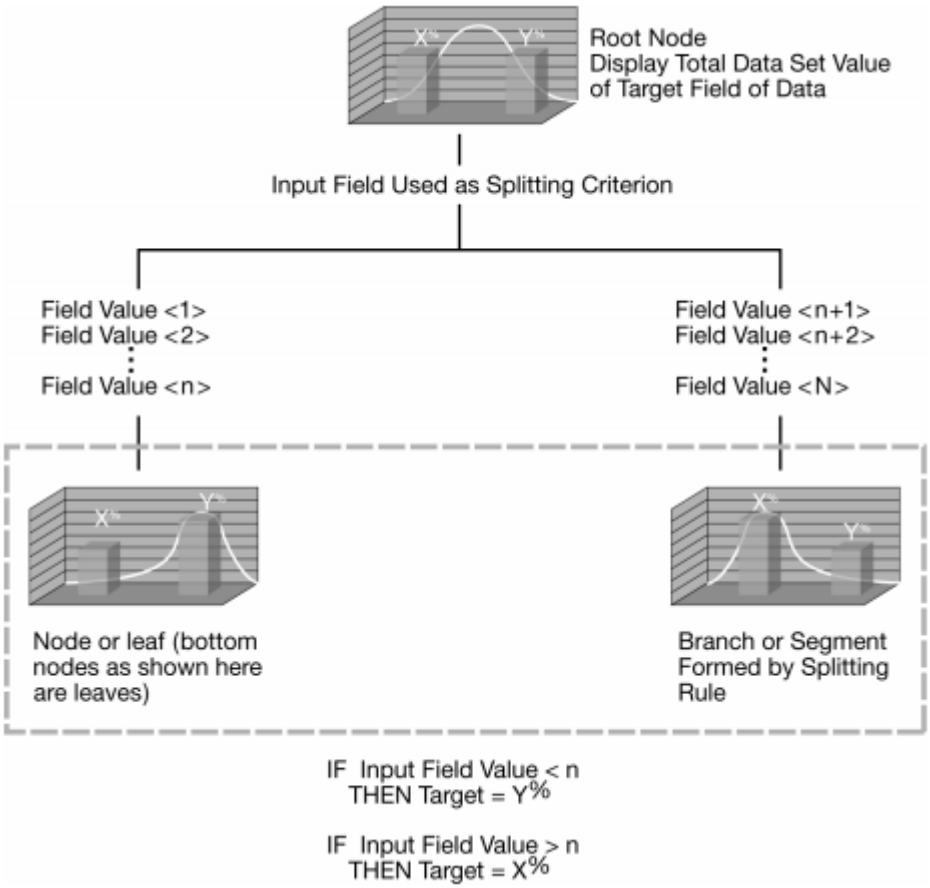
4.3.2.1. Decision Tree.

Los árboles de decisión son una forma simple, pero potente de análisis de múltiples variables. Estos algoritmos proporcionan capacidades únicas para complementar y sustituir:

- Formas de análisis estadístico tradicional como la regresión lineal múltiple.
- Una variedad de herramientas y técnicas (como las redes neuronales) de minería de datos.

Los árboles de decisión son generados por los algoritmos que identifican varias maneras de dividir un conjunto de datos establecido en segmentos con forma de árbol. Estos segmentos forman un árbol de decisión invertida que se origina con un nodo raíz en la parte superior del árbol. El objeto de análisis se representa en este nodo raíz de una forma simple y unidimensional en la interfaz de árbol de decisión. El nombre del campo que es objeto de análisis, por lo general se muestra junto con la propagación o distribución de los valores que están contenidos en ese campo.

Un ejemplo de árbol de decisión se muestra en la siguiente figura, donde se puede ver que el árbol de decisión puede reflejar tanto el análisis de datos continuos, como discretos.



En este nodo se refleja todo el conjunto de datos, registros, campos y valores de los campos que son objeto de análisis.

El descubrimiento de la regla de decisión para formar las ramas o segmentos debajo del nodo raíz se basa en un método que obtiene la relación entre el objeto de análisis y uno o más campos que sirven como campos de entrada para crear las ramas o segmentos destino. Los valores en el campo de entrada se utilizan para estimar el valor probable en el campo de destino (nodo raíz). El campo de destino, o nodo raíz, también es llamado resultado, campo dependiente o variable.

Una descripción general del enfoque de este modelado se ha representado en la imagen anterior. Una vez que se obtiene la relación, entonces una o más reglas de decisión pueden derivarse de ella para describir las relaciones entre las entradas y el resultado.

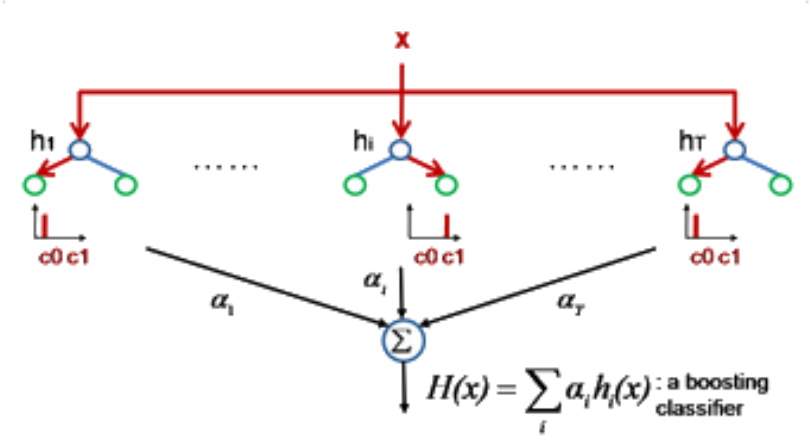
Las reglas de decisión pueden predecir los valores de observaciones nuevas o invisibles que contienen valores para las entradas, pero podría no contener valores para el resultado.

4.3.2.2. Boosting.

El algoritmo de data mining conocido como boosting se basa en generar múltiples modelos o predicciones y asignarle pesos para poder después combinar estos modelos en una única predicción o clasificación.

El funcionamiento del algoritmo de boosting se describe a continuación de una forma sencilla y básica:

Primero se aplica algún método de árbol de decisión a los datos en los que a cada observación se le asigna el mismo peso. A continuación, se calculan las clasificaciones previstas, y se aplican ponderaciones a las observaciones en la muestra de datos que son inversamente proporcionales a la exactitud de la clasificación. En otras palabras, asignar mayor peso a aquellas observaciones que son difíciles de clasificar (donde la tasa de errores de clasificación ha sido alta), y pesos menores a los que son fáciles de clasificar (donde la tasa de errores de clasificación es baja). Boosting generará una secuencia de clasificadores en los que cada clasificador consecutivo en la secuencia es un "experto" en la clasificación de las observaciones que no fueron bien clasificadas por sus predecesores. Las predicciones de los diferentes clasificadores se pueden combinar para derivar en un único modelo de predicción o clasificación mejor.



4.3.2.3. Random Forest.

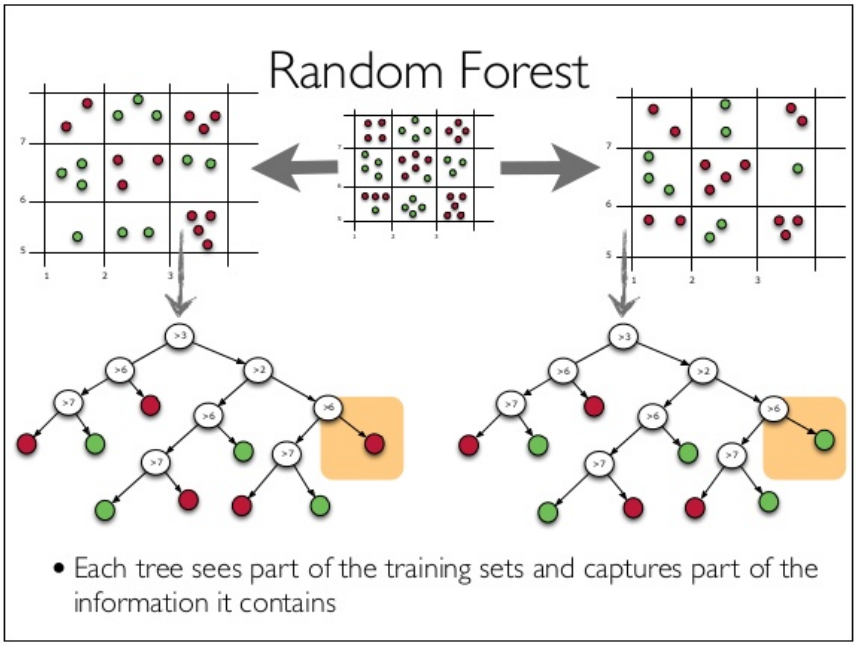
Este algoritmo viene como la evolución de otros como el boosting, descrito en el punto anterior, o el bagging.

En boosting, los sucesivos árboles dan un peso extra predicciones incorrectas realizadas en las primeras predicciones. En bagging, desaparece esta dependencia entre los sucesivos árboles, cada árbol está construido de forma independiente mediante una muestra inicial del conjunto de datos. Al final la predicción con más peso es la elegida.

Leo Breiman en 2001 propuso por primera vez el algoritmo random forest, que añade una capa de aleatoriedad a bagging. Además de construir cada árbol con una muestra inicial de los datos diferente, random forest cambia la forma en la que está construido cada árbol. En un árbol estándar cada nodo se divide usando la mejor división de entre todas las variables. En random forest, cada nodo se divide usando el mejor subconjunto el mejor de entre un subconjunto de predictores elegido al azar para ese nodo.

Esta estrategia un tanto contradictoria funciona bastante bien comparándola con otras como “support vector machine” o redes neuronales. Además es robusta frente al “overfitting”.

Además, es muy fácil de usar, ya que únicamente tiene dos parámetros que son, el número de variables en el subconjunto aleatorio, y el número de árboles del modelo, y por lo general no es demasiado sensible a los valores seleccionados.



A continuación se van a describir los pasos que sigue el algoritmo de random forest:

- Primero se seleccionan las muestras de datos de inicio de los árboles de entre los datos originales.
- Para cada una de las muestras de inicio se construye un árbol en el que en cada nodo, en lugar de elegir la mejor división entre todos los predictores, se elige una muestra al azar de entre todos los predictores y se elige la mejor división de entre estas variables.
- Después, se predicen los nuevos datos mediante la agregación de los diferentes árboles.

4.3.2.4. Linear Regresion.

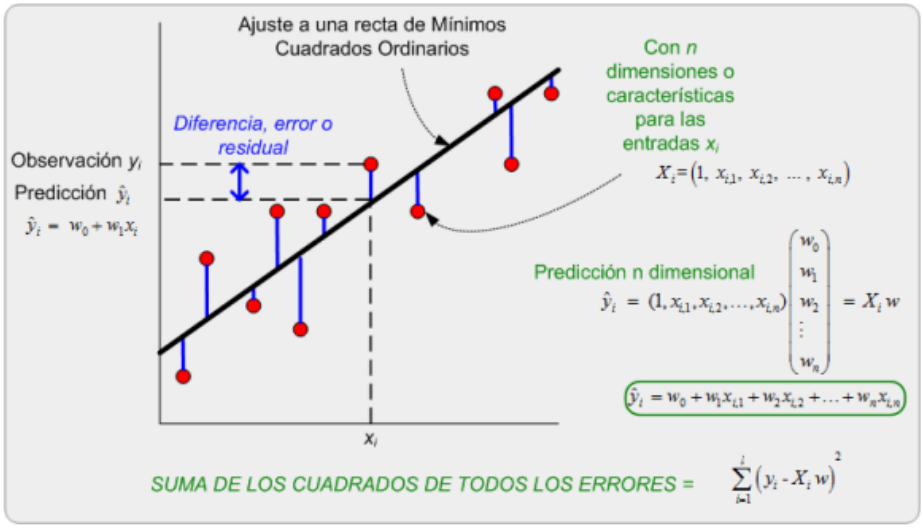
El análisis de regresión lineal es la técnica más usada en estadística y consiste en predecir el valor de una variable dependiente Y, en función de n variables independientes X.

$$\hat{Y}_t = b_0 + b_1 X_{1t} + b_2 X_{2t} + \ldots + b_k X_{kt}$$

Esta fórmula tiene la propiedad de que la predicción de Y es una función lineal de cada una de las variables X. las pendientes de cada una de estas funciones que define la relación de Y con cada X son los llamados coefcientes de las variables, representadas por las constantes b_0, b_1,…

Podemos decir por lo tanto que b_i es la predicción del cambio en el valor de Y por cada cambio en una unidad de X en igualdad de condiciones.

Generalmente las observaciones de Y (variable dependiente), son diferentes de los valores que predice la función lineal, es decir, esta tiene un error. Se define por lo tanto como la función más eficiente a aquella que tiene la menos diferencia entre los valores observados y los predichos.



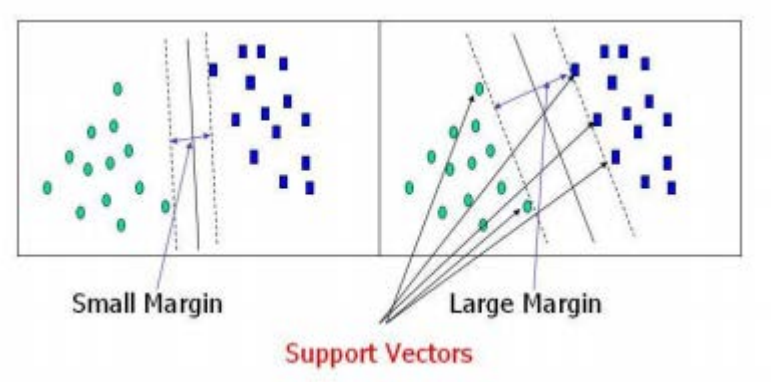
Para ajustar la función se usa lo que llaman ajuste por mínimos cuadrados, es decir, encontrar la función en la que la suma de los cuadrados de los errores (diferencia entre valor observado y valor predicho) sea la mínima posible.

4.3.2.5. Support Vector Machine.

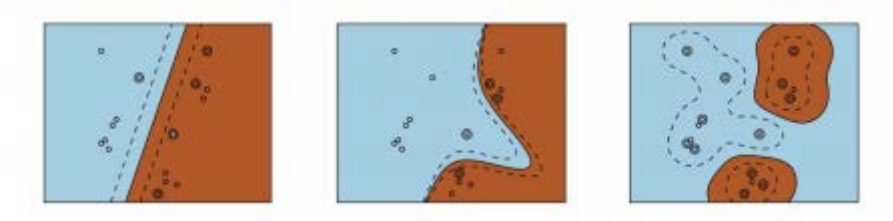
Es un popular método de clasificación binaria que trata de encontrar el hiperplano que separa los datos con el mayor margen de separación. Para ello usa un mapeo no lineal transformando los datos originales en otros de dimensión superior. En estas nuevas dimensiones, el algoritmo support vector machine busca el hiperplano óptimo que separa las clases.

Esta separación puede ser lineal o no lineal, en ambos casos, el algoritmo puede encontrar el hiperplano óptimo que separa las clases.

Un ejemplo gráfico de conjuntos de datos linealmente separables podría ser el siguiente.



A continuación un ejemplo gráfico de conjuntos de datos que no pueden ser separados linealmente, en los que el algoritmo usa ecuaciones polinómicas para resolver el problema de clasificación.



4.3.2.6. Neural Networks.

Una red neuronal son es un algoritmo complejo usado para el análisis predictivo. Está biológicamente inspirado en la estructura del cerebro humano. Una red neuronal proporciona un modelo muy simple comparado con el del cerebro humano, pero funciona bastante bien para la clasificación de datos.

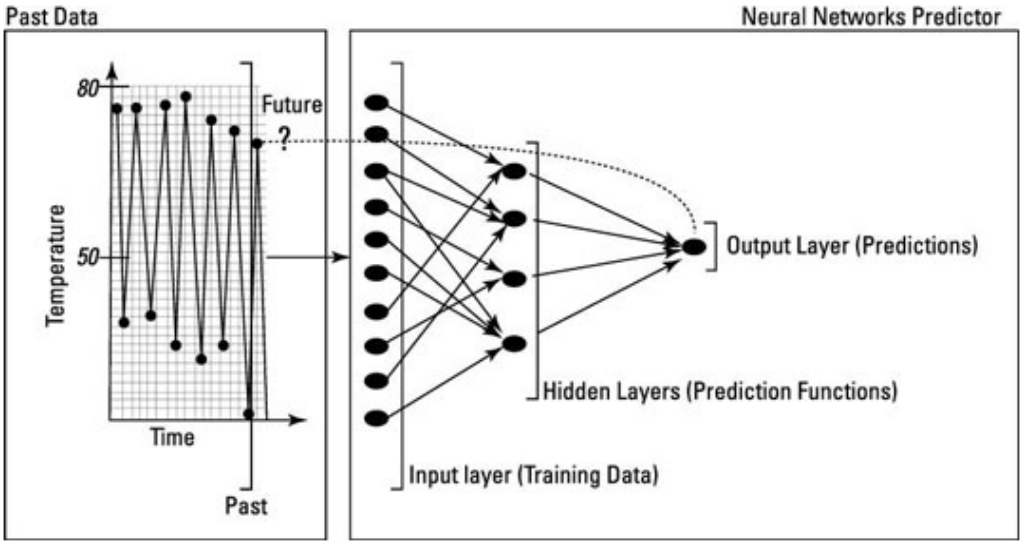
Las redes neuronales procesan datos pasado y presentes para poder realizar estimaciones de eventos futuros, descubriendo alguna correlación compleja que esté escondida en los datos, y lo hace de una forma parecida a la que emplearía el cerebro humano.

Pueden ser usadas para realizar predicciones sobre series temporales de datos, por ejemplo, datos meteorológicos. Pueden estar diseñadas para detectar

patrones en los datos de entrada y producir una salida libre de ruido o valores atípicos.

La estructura de un algoritmo de red neuronal se basa en tres capas.

- Una capa de entrada que alimenta de datos a la siguiente capa.
- Una capa oculta que contiene varias funciones complejas que crean predictores. Cada una de estas funciones se denomina neurona.
- Una capa de salida que recoge las predicciones hechas en la capa oculta y genera un resultado final.



En el ejemplo se pueden ver los datos del pasado, en este casso temperaturas recogidas a lo largo de un tiempo determinado. Los puntos negros en la primera capa es cada una de estas observaciones de temperatura, que sirven de entrada en la capa oculta para las funciones complejas o neuronas y que predicen un resultado que se muestra en la capa de salida.

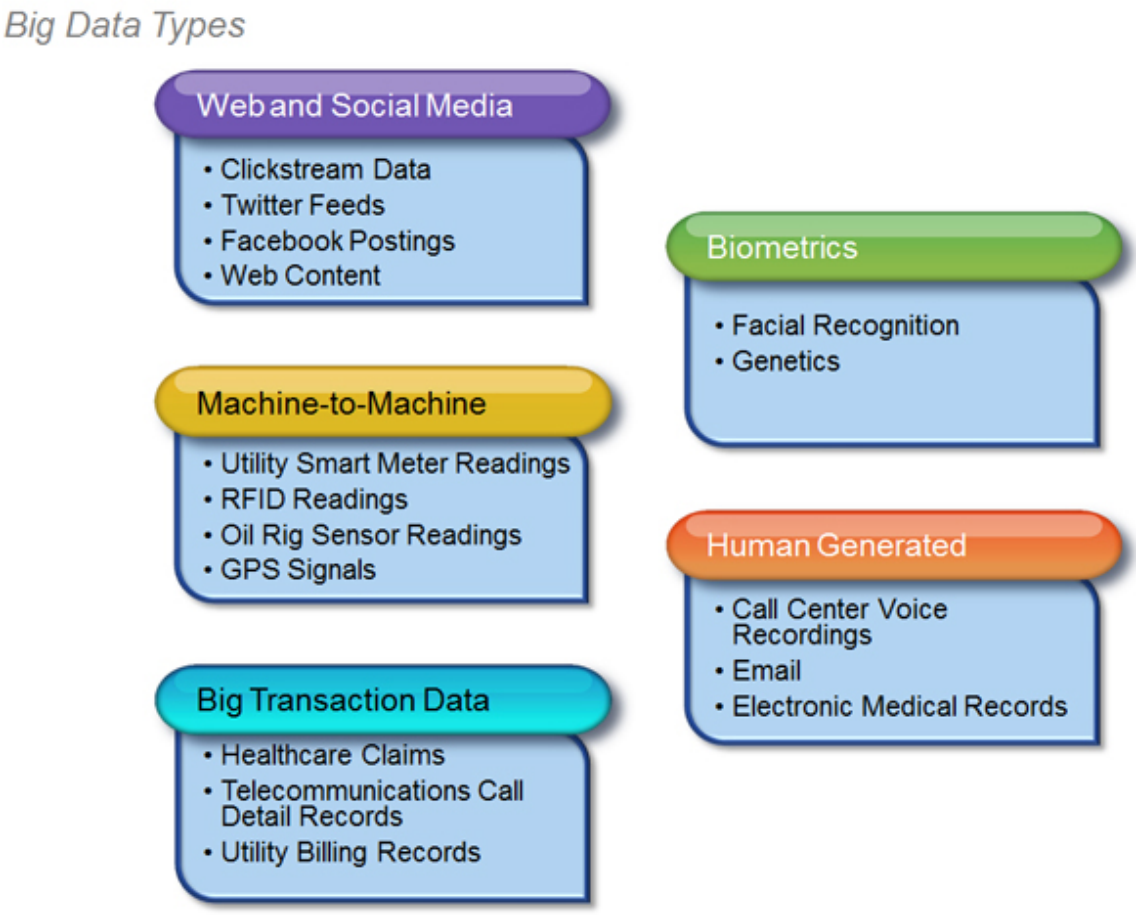
Las redes neuronales tienden a tener una alta precisión incluso aunque los datos tengan una cantidad significativa de ruido. Esto es una gran ventaja, porque al tener, la capa oculta, la capacidad de poder descubrir relaciones en los datos a pesar del ruido, hace que sea posible utilizar datos que de otro forma, no serían útiles.

Por otro lado, una de las desventajas de los algoritmos de redes neuronales es que la precisión de la predicción puede ser válida sólo dentro del período de tiempo durante el cual se recopilaron los datos.

4.4. FUENTES DEL BIG DATA.

Los sistemas de big data se basan en la obtención y el análisis de datos. Estos datos pueden ser de todo tipo y a partir de este análisis se intenta obtener un valor añadido que proporcione una ventaja competitiva frente a otros, no solo por el hecho de tener información que otros no tienen, sino de interpretarla de forma correcta.

Hoy en día las fuentes de información de las que se alimentan los sistemas de big data son muy diversas, pero se pueden clasificar tal y como se muestra en la siguiente figura.



4.4.1. Web y Social Media.

En esta categoría se podría incluir toda aquella información que se genera en las diferentes redes sociales como por ejemplo Facebook, twitter, youtube, linkedIn, Foursquare, etc…, también se pueden incluir contenidos web como blogs, wikis como la Wikipedia, agregadores de contenidos como Digg, etc…

4.4.2. Machine to Machine.

En esta categoría se incluye toda la información generada por dispositivos, sensores, o cualquier otro dispositivo tecnológico que pueda conectarse entre sí o a la red. Estos dispositivos (dispositivos GPS, sensores de presión o de velocidad, etc…) capturan información de eventos como la velocidad, la posición, la altura, la humedad, etc… y esta información es transmitida a otras aplicaciones que son las encargadas de almacenar e interpretar estos datos.

El origen de esta información es lo que anteriormente en este documento llamábamos el internet de las cosas.

4.4.3. Transaccionales.

En esta categoría se pueden incluir los datos "clásicos", aquellos que existen desde hace más tiempo y que están almacenados en bancos de datos transaccionales. Ejemplos de este tipo de información podrían ser contratos, pólizas de seguros, facturas, registros de llamadas, operaciones bancarias, etc…

Estos datos suelen estar disponibles en formato, tanto semiestructurado, como no estructurado.

4.4.4. Biometría o Reconocimiento Biométrico.

La información biométrica es aquella que se refiere a la identificación automática de una persona basada en sus características físicas. En esta categoría, por lo tanto, se incluye cualquier característica física por la que se pueda reconocer o clasificar a una persona, como por ejemplo, huellas digitales, iris, reconocimiento facial, escaneo de retina, genética, reconocimiento de voz, etc…

Los avances tecnológicos de los últimos años han aumentado considerablemente los datos biométricos disponibles.

4.4.5. Datos generados por las personas.

En esta categoría se incluye toda aquella información que genera una persona directamente, como por ejemplo, las llamadas guardadas de un call center, notas de voz, correos electrónicos, documentos electrónicos, estudios, registros médicos, faxes, etc…

Uno de los problemas más habituales de este tipo de información es que suele contener información sensible. Para solucionarlo, normalmente debe ser ocultada, o cifrada de alguna forma de cara a conservar la privacidad de las personas y evitar problemas con la ley.

4.5. SEGURIDAD.

Hoy en día, los datos son un activo más de las empresas. Estos datos, como hemos estado viendo pueden ser analizados en busca de un beneficio, ya sea por describir e informar cómo ha funcionado la empresa hasta ese día, que caminos se han seguido, que decisiones han sido buenas o malas en función del resultado que han proporcionado, o para intentar predecir que decisiones van a ser las mejores en un futuro, intentando predecir cuál será la mejor decisión, prediciendo su resultados. Todo esto vale mucho dinero para una empresa y por tanto, las empresas tienen mucho cuidado con estos datos e intentar ponerlos a buen recaudo, darles la mayor seguridad posible.

Para proporcionar a estos datos de seguridad existen varios sistemas entre los que podemos destacar los tres siguientes:

4.5.1. Information Rights Management (IRM).

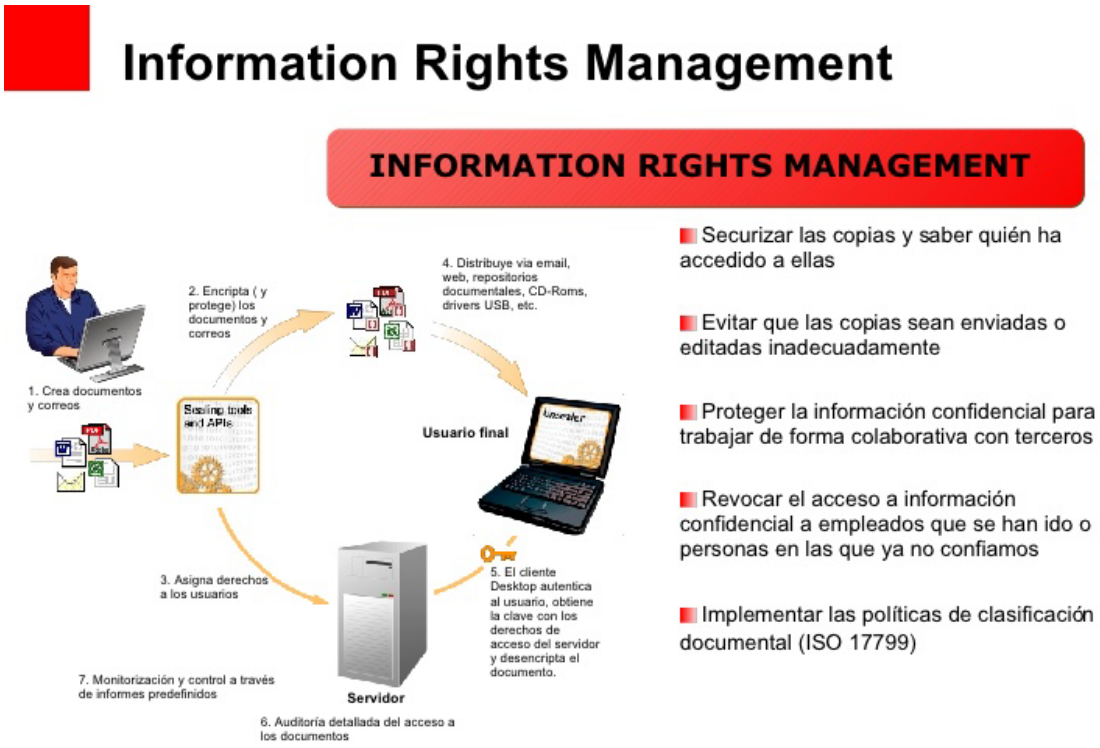
Son un conjunto de técnicas y métodos destinados a proteger información sensible de accesos no autorizados. Permiten que la información (la mayor parte de ella) pueda ser controlada remotamente. Esto significa que el control de la información puede realizarse de forma separada para las diferentes acciones del usuario como la creación, edición, consulta y/o distribución de los datos.

Estas tecnologías usan el cifrado para prevenir el acceso no autorizado. El acceso a los datos cifrados puede realizarse únicamente a través de una password o clave.

Una vez que la información está cifrada, el sistema puede aplicar ciertos permisos de acceso que le darán a un usuario la posibilidad de realizar o no ciertas acciones sobre un conjunto de información. Algunos de estos permisos son:

- Protección sobre el uso de los datos, como por ejemplo, copiar y pegar, impedir capturas de pantalla, impresión o edición de los datos.
- Creación de un modelo o “policy” que permita realizar de forma sencilla un mapeo de clasificaciones de negocio a la información.
- Auditoría completa de tanto el acceso a los datos, así como de los cambios en los permisos de acceso, edición, consulta o distribución de los mismos.

En la siguiente figura se puede ver un ejemplo del funcionamiento de un sistema de gestión de los derechos de la información (IRM).



4.5.2. Data Loss Prevention (DLP).

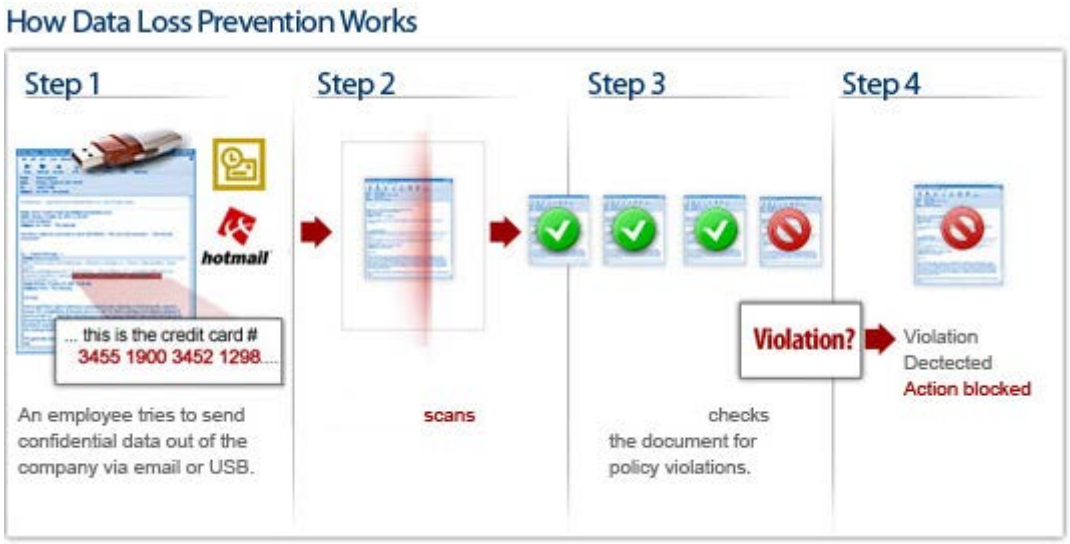
La tecnología Data Loss Prevention, también conocida como Data Leak describe sistemas y tecnologías diseñadas para detectar posibles violaciones de acceso a información, o intentos de mover información fuera de la seguridad de los sistemas de almacenamiento de una organización. Por prevención se entiende la supervisión de los sistemas y la detección y el bloqueo de accesos o transmisiones de información confidencial.

En general los sistemas de DLP proporcionan tres tipos distintos de protección:

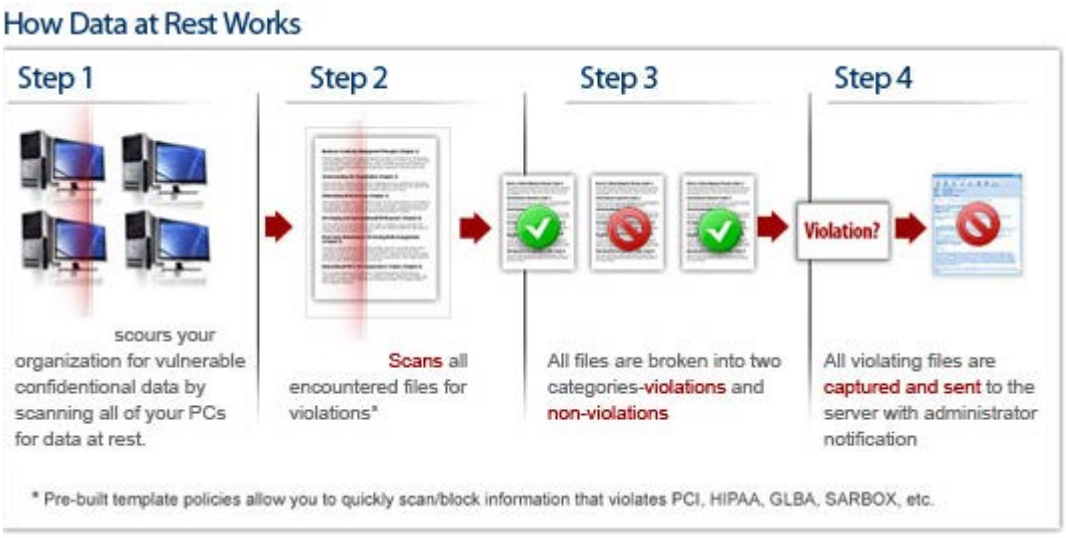
- Uso de los datos (In-Use Protection). Esta protección aplica cuando los datos están en uso ya sea por aplicaciones o por usuarios. Se basa en varios tipos de autenticación de usuario para establecer la identidad de los que están solicitando el acceso a los datos, junto con sistemas de control de acceso que permiten o no el acceso a los datos en función de la identidad del usuario, de su rol y de las políticas de seguridad implantadas.

Generalmente, además los datos permanecen cifrados en todo momento por lo que aunque se pueda acceder a archivos de trabajo temporales o a partes sueltas de la memoria, no se podría ver nada de información relevante en ningún momento.

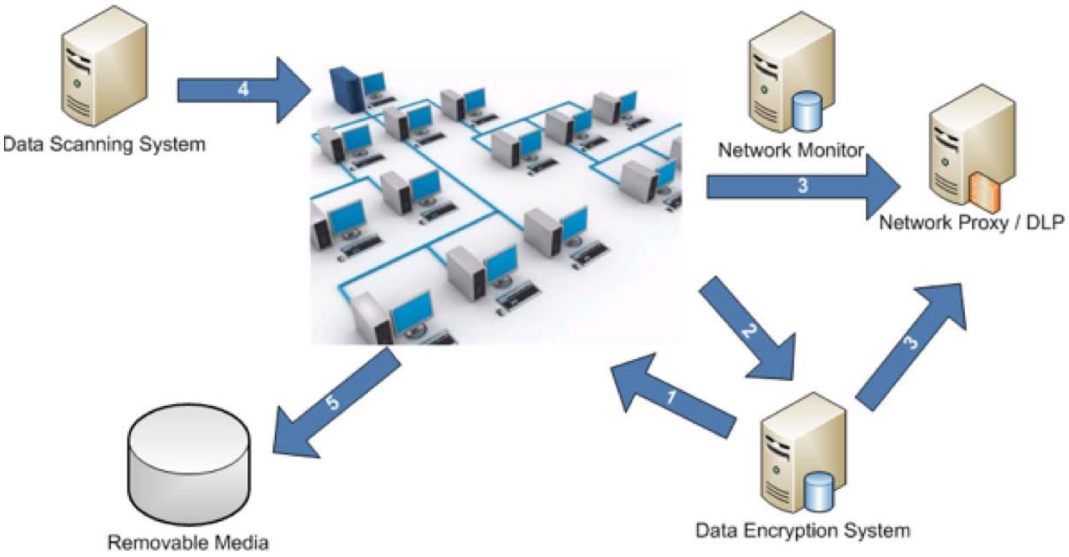
- Transmisión de los datos (In-motion protection). Esta protección aplica cuando los datos están en movimientos, transmitiéndose a través de una red de cualquier tipo. En general, este tipo de protección depende mucho de que las técnicas y tecnologías de cifrado sean lo suficientemente fuertes para mitigar el riesgo de espionaje y reducir considerablemente la posibilidad de un ataque de descifrado con éxito. Cuanto más valioso sea el dato, más fuerte debería ser el cifrado.



- Almacenamiento de datos (Data in-rest). Esta protección aplica a aquellos datos que están almacenados de alguna forma en algún medio de almacenamiento persistente. Este tipo de protección implica generalmente, controles de acceso de cara a limitar accesos exclusivamente a aquellos que tengan permiso. Monitorizar accesos para registrar y poder rastrear cualquier acceso a la información. Y cifrado de los datos para proteger el sistema del robo físico del medio de almacenamiento de los datos.



En el siguiente diagrama se muestra un sistema de control a modo de ejemplo. Es un sistema que se encarga de examinar el flujo de información dentro y fuera de la red, y que intenta minimizar el riesgo de pérdida o violación de datos a través de dispositivos de almacenamiento extraíbles.



4.5.3. Database Activity Monitoring (DAM).

Es una tecnología de seguridad de base de datos que sirve para monitorizar y analizar la actividad de la base de datos y que trabaja independiente del sistema gestor de base de datos (DBMS), es decir, no tiene ninguna relación con la auditoría interna del gestor de base de datos ni con sus log transaccionales.

A continuación se describen varios casos de uso típicos para un Database Activity Monitoring:

- Monitorización de usuarios privilegiados o superusuarios.

  Estos usuario pueden ser por ejemplo los administradores de la base de datos, administradores del sistema, desarrolladores, etc… que suelen tener acceso sin restricciones a las bases de datos corporativas.

  La monitorización de estos usuarios incluye:

  - Auditoría de toda la actividad y transacciones realizadas por el usuario.
  - Identificación de actividad anómala, como por ejemplo, la visualización de información sensible o la creación de cuentas con permisos de superusuario.
  - Conciliación de actividades con peticiones de cambio autorizadas como por ejemplo crear o borrar tablas.

  La mayor parte de las empresas están protegidas de forma perimetral, sin embargo, también es muy importante la protección frente a ataques internos como los que pueden llevar a cabo usuarios con privilegios de superusuario y que son muy difíciles de detectar mediante las técnicas tradicionales.

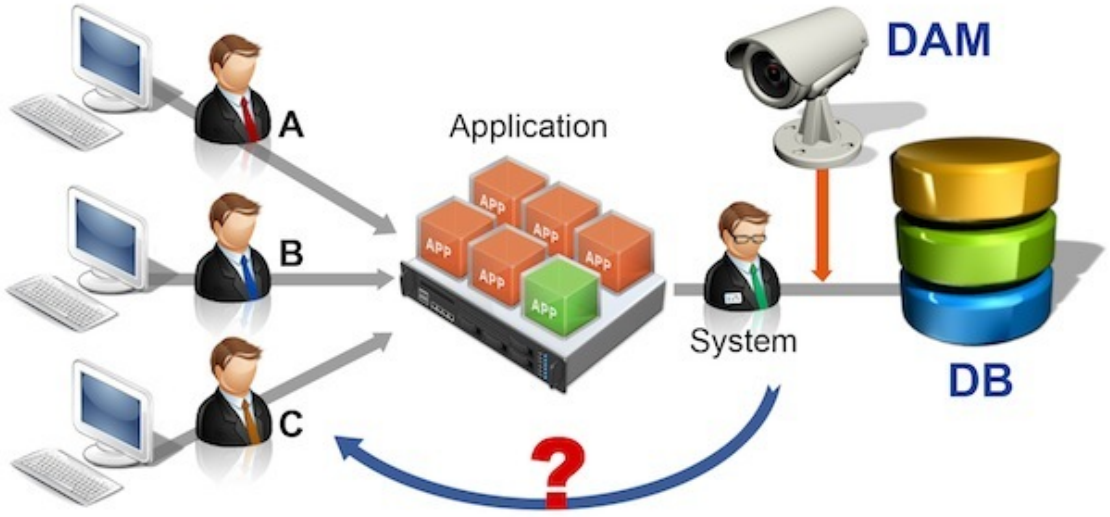
  Además, como la mayor parte de los ataques se basan en conseguir controlar un usuario con permisos de superusuario, el seguimiento de las actividades de estos usuarios puede ayudar a encontrar de forma eficaz los sistemas comprometidos.

- Monitorización de la actividad de las aplicaciones.

  El propósito principal de la monitorización de la actividad de las aplicaciones que acceden a la base de datos es el de proporcionar un mayor nivel de responsabilidad del usuario final y detectar el fraude u otros accesos ilegítimos que se producen a través de software de la empresa en vez de mediante accesos directos a la base de datos.

  Existe software que realiza conexiones y accesos a la base de datos ocultando la identidad de los usuarios finales, ya que se basan en lo que llaman agrupación de conexiones, esto es, utilizan conexiones agrupadas, la aplicación agrega todo el tráfico dentro de unas pocas conexiones a la base de datos que se identifican únicamente por el nombre de una cuenta de servicio genérica. La monitorización de la actividad de aplicaciones permite a la organización

asociar cada conexión a la base de datos con un usuario final con el fin de identificar las actividades no autorizadas o sospechosas.



El uso de DAM está impulsado principalmente por la necesidad de control sobre los usuarios con privilegios de superusuario, aunque las necesidades de las empresas están provocando que la utilidad de DAM se extienda más allá de las funciones básicas tales como la capacidad de detectar actividades maliciosas o la administración inadecuada de la base de datos o el acceso no autorizado a la misma.

Otras funciones más avanzadas también incluidas en las tecnologías DAM son:

- La capacidad de monitorizar los ataques dentro de la base de datos y las puertas traseras en tiempo real.
- El bloqueo y la prevención de transacciones sin estar conectado.
- Descubrimiento activo de los datos en riesgo.
- Mejora la visibilidad del tráfico de las aplicaciones.
- Capacidad para ofrecer monitorización de la actividad de la base de datos en entornos virtualizados o incluso en la nube.

Existen tres tipos comunes de arquitectura para los sistemas DAM:

- Interception Based.
- Memory Based.
- Log Based.

4.6. VISUALIZACIÓN.

Otra parte muy importante del estudio del big data es cómo entender los resultados de los análisis del big data. De este campo se ocupan las diferentes herramientas y técnicas de visualización del big data que ayudan a los usuarios a crear gráficos en dos y en tres dimensiones (a veces incluso cuatro) que ayuden a entender los conjuntos de datos del negocio y a tener una mejor visión interna del mismo.

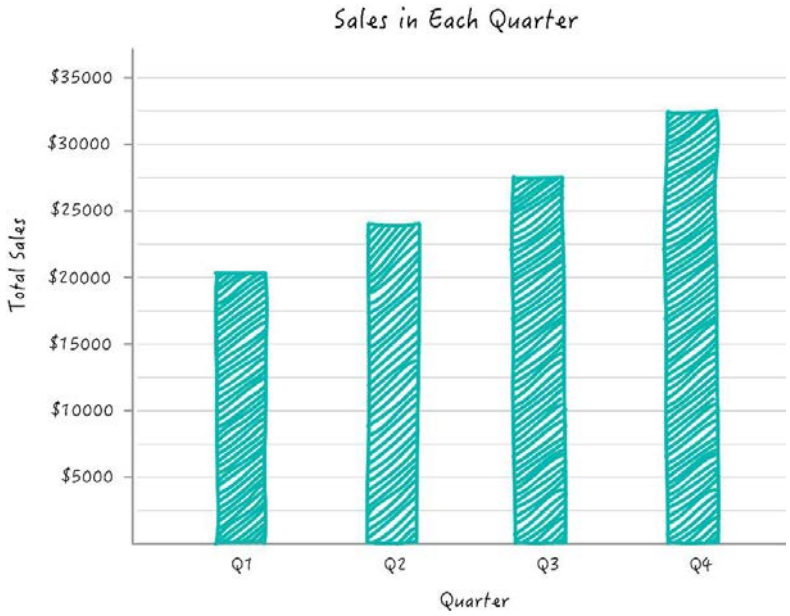
La forma en la que se representen los datos, por lo tanto, se convierte en una parte importante del análisis de estos datos, ya que a través de la representación de los datos se puede conseguir que éstos se entiendan de una forma clara, eficiente y rápida.

Para poder realizar esta elección de cómo representar los datos, Carolina Cristanchi, creo la siguiente ilustración que propone un tipo de grafico u otro en función de la finalidad que se tiene al representar los datos.

A continuación se van a detallar algunos de estos gráficos en función de su finalidad:

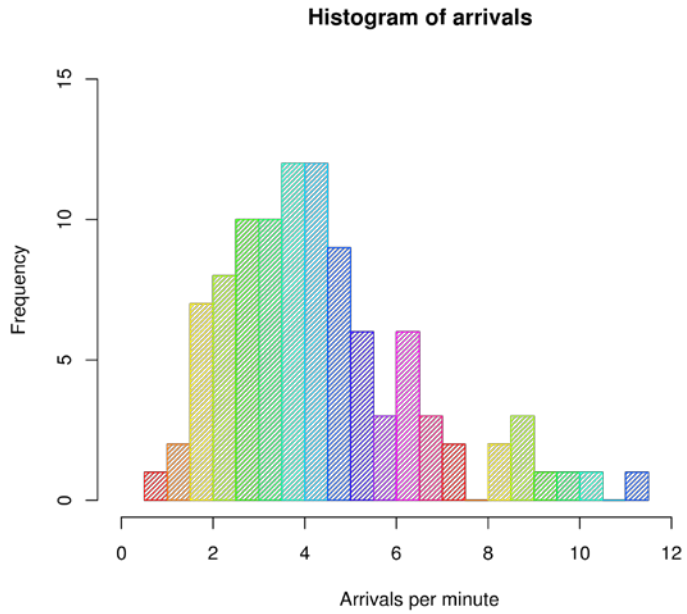
4.6.1. Comparación.

Como ejemplo para la comparación de diferentes datos, tenemos los gráficos de barras, que a pesar de ser sencillos, permiten comparar magnitudes de varias categorías, o expresar la evolución de una en el tiempo



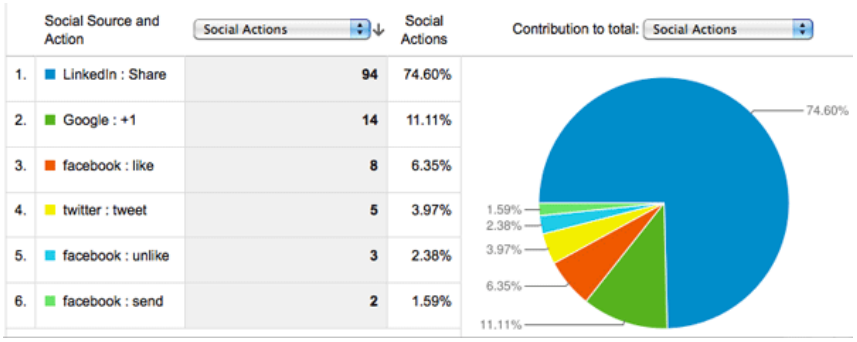
4.6.2. Distribución.

Para representar una variable cuantitativa continua es muy útil el histograma, en el cual, la superficie de cada barra es proporcional a la frecuencia de los valores representados.



4.6.3. Composición.

Si lo que se desea representar son las magnitudes de las diferentes partes de un todo, gráficos como el de tartas, aunque si el número de categorías es demasiado elevado, pueden llegar a ser confusos e ilegibles.



	Social Source and Action	Social Actions	Social Actions
1.	LinkedIn : Share	94	74.60%
2.	Google : +1	14	11.11%
3.	facebook : like	8	6.35%
4.	twitter : tweet	5	3.97%
5.	facebook : unlike	3	2.38%
6.	facebook : send	2	1.59%

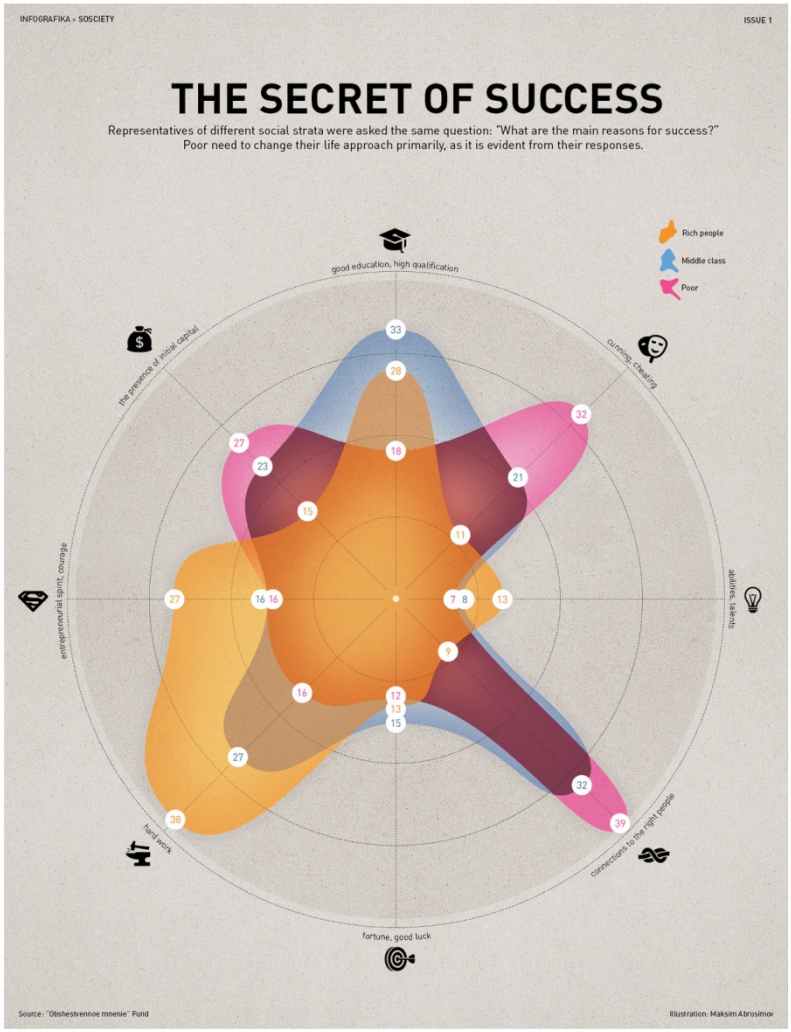
Contribution to total: Social Actions

Para aquellos casos en los que las categorías son demasiadas para ser representadas en un gráfico de tartas de forma clara, se tiene la posibilidad del denominado mapa de árbol. En esta representación gráfica se puede visualizar la división entre áreas proporcionales a los datos y con una jerarquía tanto por tamaños, como por colores. Pueden incluso tener los porcentajes y una pequeña leyenda para cada representación.

Un ejemplo muy buena de este tipo de representación es el del MIT, https://atlas.media.mit.edu/es/explore/tree_map/hs/export/show/all/5201/2009/.

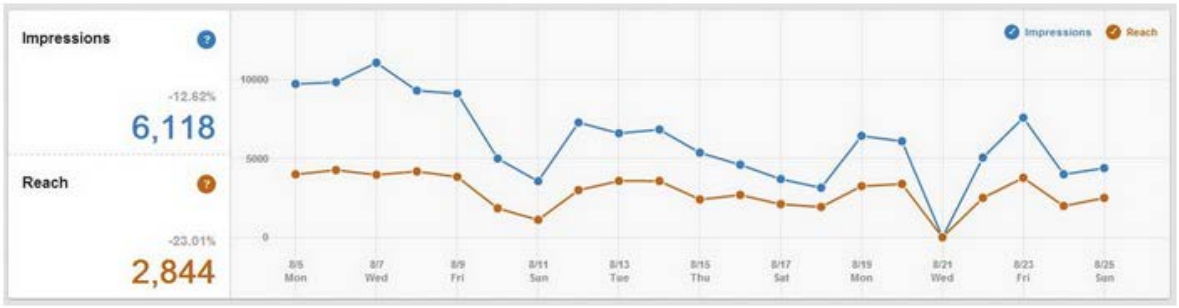
4.6.4. Relación.

Para poder representar la dirección o tendencia al comparar diversas variables el mejor ejemplo es el gráfico de radar, como por ejemplo esta infografía de StatsChat.org del secreto del éxito comparando los diferentes motivos del éxito en función del estrato social al que perteneces.



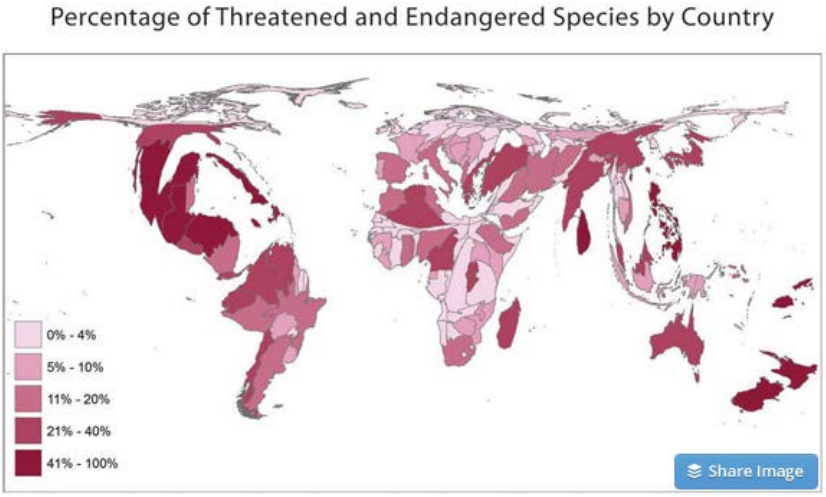
4.6.5. Evolución.

Para mostrar cambios a los largo del tiempo mientras se relacionan dos o más variables, el mejor gráfico es el lineal, como en este ejemplo en el que se pueden ver el alcance y las impresiones en pinterest.



4.6.6. Geografía.

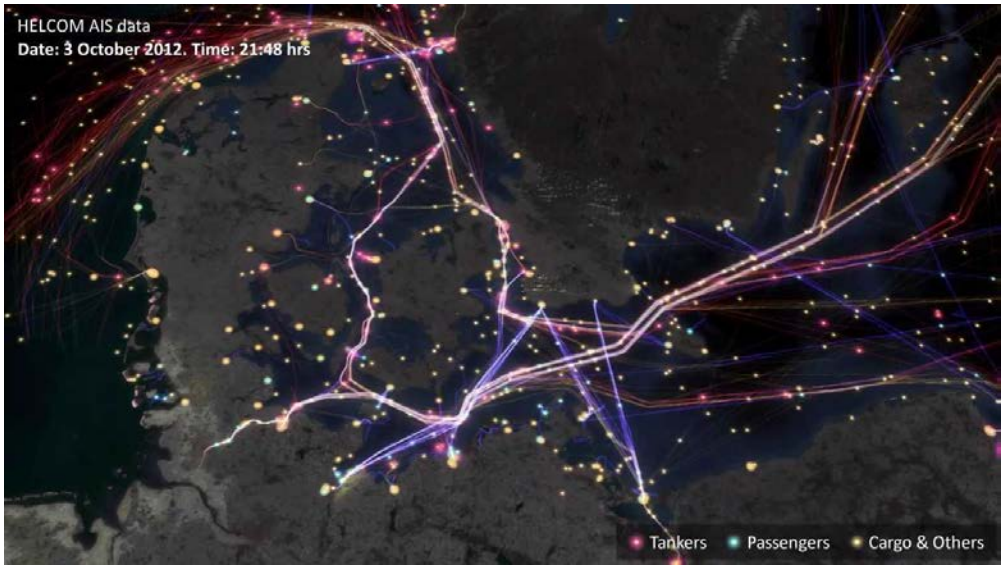
Si se quiere representar datos por regiones, la mejor representación son los cartogramas.



En algunos casos, la unión de este tipo de representaciones geográficas con la componente de evolución, la componente temporal, consigue algunos de los ejemplos más llamativos de representación de grandes cantidades de información a través de videos. Por ejemplo, con los datos de los vuelos realizados en el mundo en 24 horas, se montó el siguiente video de un minuto en el que se pueden apreciar fácilmente las rutas más usadas y en que horarios se vuela más en cada parte del mundo.

https://www.youtube.com/watch?v=yx7_yzypm5w

El siguiente paso es incluir también la componente de relación. En el siguiente ejemplo se muestra geovisualmente el tráfico marítimo existente en el mar Báltico relacionándolo con el número de accidentes que causan el vertido de petróleo al mar.



https://www.youtube.com/watch?v=X9UtUzHDn4c

5. CONCLUSIONES.

Es obvio que en los últimos años los datos generados en el mundo están aumentando exponencialmente. Es evidente también, que el conocimiento de esta información, y sobre todo el análisis de ésta, cuyo objetivo es obtener información nueva a través de la interpretación de los datos existentes, de cara a obtener un beneficio, se ha convertido en uno de los principales objetivos de las organizaciones de hoy en día.

La consecución de este objetivo se da gracias a la mejora continua del hardware, que ha provocado la posibilidad de tener máquinas baratas y con gran potencia, y a la aparición de un conjunto de herramientas dedicadas a la gestión y análisis de grandes volúmenes de datos, comúnmente llamadas “Big Data”.

Big Data permite, por ejemplo, obtener una imagen más completa de las preferencias y demandas de los clientes, y las empresas están empezando a invertir mucho en el análisis y la comprensión del Big Data para encontrar nuevas formas de interactuar con sus clientes actuales y futuros. Pero tiene muchos más usos o aplicaciones como el análisis de riesgos para la mejora de la seguridad, el análisis de operaciones de cara a la optimización operativa o a la búsqueda de un nuevo modelo empresarial.

Sin embargo, cuando se usan herramientas de Big Data para la búsqueda de estos objetivos, una de las decisiones más importantes es precisamente la elección de la mejor tecnología, ya que una mala elección es sinónimo de fracaso. Es por esto, por lo que se deben realizar estudios detallados que aumenten el conocimiento de todas las herramientas y tecnologías existentes, ya sea mediante el estudio de landscapes existentes, o como en este proyecto, de la creación de otros originales que permitan identificar mejor las características de estas herramientas, de cara a realizar mejor elección para asegurar el éxito

6. PRESUPUESTO.

A continuación se detalla el presupuesto del proyecto, en él se van a tener en cuenta todos os elementos que han sido necesarios para realizar el mismo.

La realización de este presupuesto se ha realizado considerando una amortización de tres años y teniendo en cuenta una dedicación al proyecto de un año. Los dos años restantes, los elementos adquiridos se usarán para la realización de otros proyectos por lo que el coste aplicable de cada uno de estos elementos a este proyecto es de un tercio de su valor amortizado.

La amortización se ha realizado con los siguientes elementos adquiridos para la realización del proyecto:

Portatil VOSTRO 3360 I5-3317U 4/500 13.3"	714,93€
Impresora HP LaserJet Pro 200 color MFP M276nw	330,48€
Paquete Microsoft Office Hogar y Empresas 2013	269,00€
TOTAL	**1314.41€**

Todos los precios mostrados en la tabla incluyen el IVA.

Las condiciones del préstamo solicitado para financiar estos elementos son las que se detallan a continuación.

TAE	7,18%
TIN	6,95%
Plazo	3 años
Comisión Apertura	2%
Comisión Estudio	0%

Teniendo en cuenta estas condiciones, la amortización queda como sigue.

Importe del crédito	1315,00€
Plazo	3 años
Plazo correspondiente al proyecto	1 año
Cuota mensual	40,70€
Gastos Totales*	1491,50€
Intereses	176,50€
GASTO APLICABLE AL PROYECTO	**497,17€**

* Cuotas más comisiones y gastos

Este gasto desglosado por elementos queda, por lo tanto, tal y como aparece en la siguiente tabla.

Elemento	Coste Total	Coste proyecto
Portatil VOSTRO 3360 I5-3317U 4/500 13.3"	714,93€	270,42€
Impresora HP LaserJet Pro 200 color MFP M276nw	330,48€	125,00€
Paquete Microsoft Office Hogar y Empresas 2013	269,00€	101,75€
TOTAL	1314,41€	497,17€

Por último se detalla en la siguiente página el presupuesto final del proyecto.

PRESUPUESTO PFC

	Unidades	Coste por Unidad		Coste Total
Hardware				
Portatil Vostro	1	270,42 €		270,42 €
Impresora HP	1	125,00 €		125,00 €
Pendrive Sandisk	1	44,00 €		44,00 €
			Total Hardware	**439,42 €**
Software				
Microsoft Office	1	101,75 €		101,75 €
			Total Software	**101,75 €**
Recursos Humanos				
Ingeniero Informático	920	69,24 €		63.700,80 €
			Total RRHH	**63.700,80 €**
Documentación				
Conexión a Internet	12	19,90 €		238,80 €
			Total Documentación	**238,80 €**
Consumibles				
Papel (paquete de 500 folios)	1	3,00 €		3,00 €
Encuadernación	2	3,50 €		7,00 €
Varios	1	45,00 €		45,00 €
			Total Consumibles	**55,00 €**

TOTAL PFC 64.535,77 €

7. BIBLIOGRAFÍA

[1] Dumbill, E. (2012). *Radar O´Reilly*. Obtenido de http://radar.oreilly.com/2012/01/what-is-big-data.html

[2] CISCO. (2014). *www.cisco.com.* Obtenido de http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

[3] Kranenburg, R. v. (2011). *http://www.researchgate.net/.* Obtenido de http://www.researchgate.net/profile/Matt_Ratto/publication/228360933_The_Internet_of_Things/links/0912f513755ebd1e87000000.pdf

[4] Ghemawat, S. G. (2003). *research.google.com.* Obtenido de http://static.googleusercontent.com/media/research.google.com/es//archive/gfs-sosp2003.pdf

[5] Deam, J. G. (2004). *research.google.com.* Obtenido de http://static.googleusercontent.com/media/research.google.com/es//archive/mapreduce-osdi04.pdf

[6] Bodas Sagi, D. J. (2014). Introducción al paradigma del big data. Centro Universitario de Tecnología y Arte Digital.

[7] Davis, J. (2012). *http://blogs.sas.com*. Obtenido de http://blogs.sas.com/content/corneroffice/2012/10/08/what-kind-of-big-data-problem-do-you-have/

Apache.org. (s.f.). *Cassandra*. Obtenido de http://cassandra.apache.org/

Apache.org. (s.f.). *Hadoop*. Obtenido de https://hadoop.apache.org/

Apache.org. (s.f.). *HBase.* Obtenido de http://hbase.apache.org/ & http://hbase.apache.org/book.html

Apache.org. (s.f.). *Mahout.* Obtenido de http://mahout.apache.org/

Barranco Fragoso, R. (2012). *http://www.ibm.com.* Obtenido de http://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/

Cárdenas Montes, M. (2014). *http://wwwae.ciemat.es.* Obtenido de http://wwwae.ciemat.es/~cardenas/docs/curso_MD/svm.pdf

Datastax. (2015). *http://docs.datastax.com/.* Obtenido de http://docs.datastax.com/en/cassandra/2.0/pdf/cassandra20.pdf

Gkavresis, G. (s.f.). *HBASE.* Obtenido de The Distributed Management of Data Laboratory: http://www.dmod.eu/WeeklyMeeting/hbase.pdf

HortonWorks. (s.f.). *http://hortonworks.com*. Obtenido de http://hortonworks.com/hadoop/mahout/

IBM. (2009). *https://www.ibm.com*. Obtenido de https://www.ibm.com/developerworks/java/library/j-mahout/index.html

Keahey, T. A. (2014). *http://dataconomy.com.* Obtenido de http://dataconomy.com/wp-content/uploads/2014/06/IBM-WP_Using-vis-to-understand-big-data.pdf

Leo Breiman, A. C. (s.f.). *http://www.stat.berkeley.edu.* Obtenido de http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Martinez Casas, D. (2013). Taller Apache Cassandra. *Curso Big Data* . Universidad de Santiago de Compostela.

Oracle. (s.f.). *http://www.oracle.com.* Obtenido de http://www.oracle.com/technetwork/database/nosqldb/learnmore/nosql-database-498041.pdf

Saiz, C. (2015). *http://www.analiticaweb.es.* Obtenido de http://www.analiticaweb.es/dataviz-o-visualizacion-de-datos-primeros-pasos/

SAS. (s.f.). *http://support.sas.com.* Obtenido de http://support.sas.com/publishing/pubcat/chaps/57587.pdf

SourceForge. (2013). *NewSQL*. Obtenido de http://newsql.sourceforge.net/